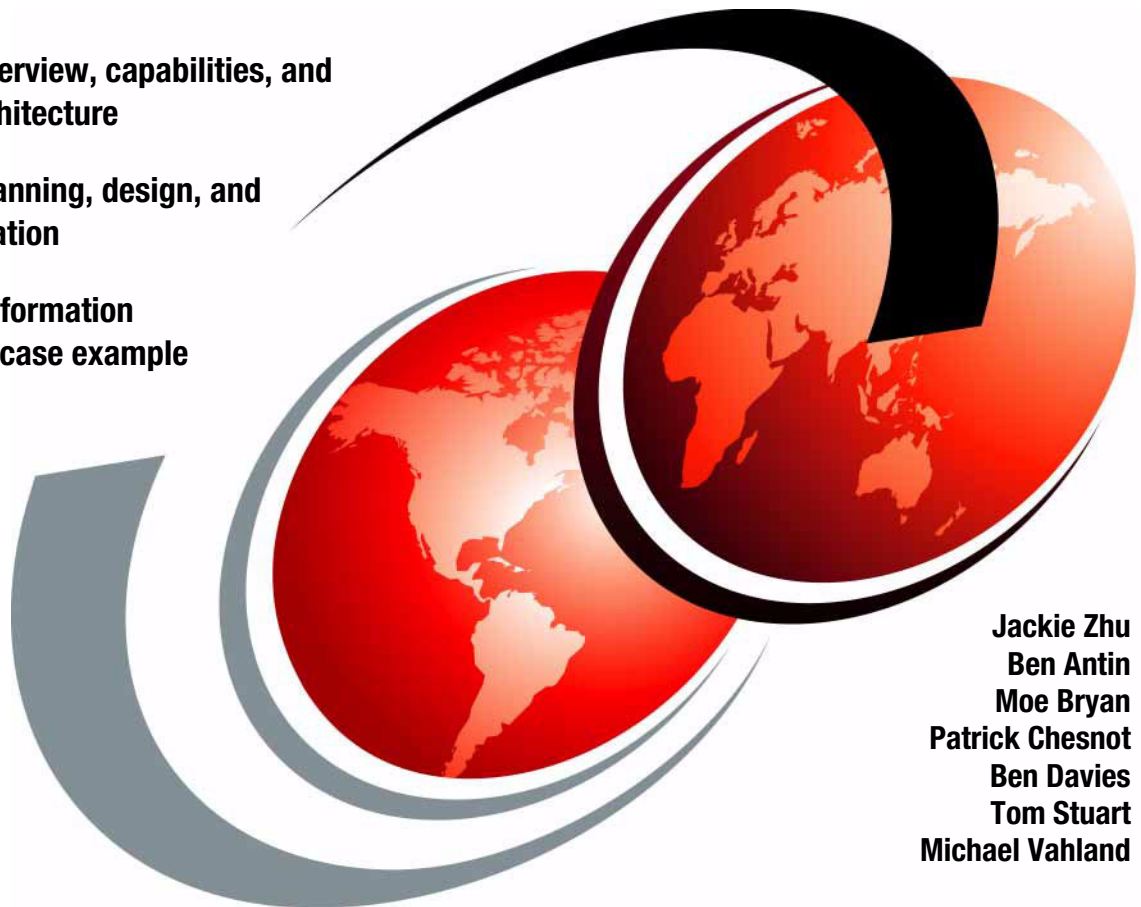


Implementing Imaging Solutions with IBM Production Imaging Edition and IBM Datacap Taskmaster Capture

Solution overview, capabilities, and system architecture

Solution planning, design, and implementation

Practical information with a use-case example



Jackie Zhu
Ben Antin
Moe Bryan
Patrick Chesnot
Ben Davies
Tom Stuart
Michael Vahland



International Technical Support Organization

**Implementing Imaging Solutions with
IBM Production Imaging Edition and
IBM Datacap Taskmaster Capture**

October 2011

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (October 2011)

This edition applies to Version 5, Release 0, of IBM Production Imaging Edition (product number 5725-B54) and Version 8, Release 0, of IBM Datacap Taskmaster Capture.

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team who wrote this book	xiv
Now you can become a published author, too!	xvi
Comments welcome	xvi
Stay connected to IBM Redbooks	xvii
Part 1. Production imaging	1
Chapter 1. Production imaging overview	3
1.1 The business document problem and approach to its solution	4
1.1.1 Paper everywhere	4
1.1.2 Business challenges posed by paper	5
1.1.3 Business challenges posed by electronic documents	6
1.1.4 Solving the problem with production imaging	7
1.2 Introduction to Production Imaging Edition	10
1.2.1 Components of the Production Imaging Edition offering	11
1.2.2 Taskmaster base and add-on components	13
1.2.3 The production imaging process	15
1.2.4 Focus on Taskmaster	21
1.3 Examples of applications	22
1.3.1 Cross industry: Automated forms processing	23
1.3.2 Cross industry: Distributed capture	23
1.3.3 Cross industry: General business documents processing	24
1.3.4 Cross industry: Accounts payable	25
1.3.5 Cross industry: Surveys	27
1.3.6 Government: Tax return processing	28
1.3.7 Healthcare and insurance: Medical claims	29
1.3.8 Banking and finance: Loan applications	32
1.3.9 Transportation and logistics: Shipping documents	32
1.4 Conclusion	33
Chapter 2. System architecture	35
2.1 Architecture overview of Production Imaging Edition	36
2.2 Components of the Taskmaster system	37
2.3 Components of FileNet Content Manager	41
2.4 Overall system architecture	45

2.5	Deployment of Production Imaging Edition	46
2.5.1	Centralized deployment	46
2.5.2	Distributed deployment	47
2.5.3	Taskmaster Web deployment	49
2.6	Conclusion.	49
Chapter 3.	Production imaging functionality	51
3.1	Functionality highlights of Taskmaster	52
3.2	Taskmaster process	53
3.2.1	Batch preparation for scanning	54
3.2.2	The Scan task	55
3.2.3	Background processing task	55
3.2.4	The Verify task	56
3.2.5	The Export task	56
3.3	Taskmaster GUI	56
3.3.1	Productive GUI	57
3.3.2	Image snippets	57
3.3.3	Color-coded recognition confidence	57
3.3.4	Click'n'Key capability.	58
3.4	Taskmaster clients	58
3.4.1	Taskmaster Client (thick client)	58
3.4.2	Taskmaster Web client	68
3.5	Taskmaster background processes	71
3.5.1	Rule processing	71
3.5.2	Job, task, and task profile	73
3.5.3	Rule set.	73
3.5.4	Rule.	74
3.5.5	Processing of the document hierarchy at run time	74
3.5.6	Function and action.	75
3.6	Taskmaster action libraries	76
3.6.1	Image cleanup and enhancements	80
3.6.2	Barcode recognition	81
3.6.3	Optical Character Recognition	82
3.6.4	Intelligent Character Recognition	84
3.6.5	Optical Mark Recognition	84
3.6.6	Classification	86
3.6.7	Fingerprinting	87
3.6.8	Content-based identification with IBM Classification Module	87
3.6.9	Language support	88
3.6.10	Imprinting and redaction	89
3.6.11	Locating text	90
3.6.12	Validations.	90
3.6.13	Exports	91

3.7 Principles and tools of the Taskmaster configuration	94
3.7.1 Datacap Studio	96
3.7.2 Flex Capture and Flex Manager	99
3.7.3 Taskmaster Application Manager	101
3.7.4 RV2 report viewer	103
3.7.5 NENU	103
3.8 FileNet Content Manager for production imaging	105
3.8.1 Workflow management tools	106
3.9 Advanced production imaging viewing	113
3.9.1 PDF viewing and annotating	114
3.9.2 Universal viewing and annotating	115
3.9.3 Document streaming	115
3.9.4 Permanent redaction	116
3.10 Bulk Import Tool	117
3.11 Conclusion	119
Part 2. Solution implementation	121
Chapter 4. Solution example	123
4.1 Scenario background	124
4.2 Current claim approval process	125
4.3 New claim approval process	128
4.4 Summary of benefits	136
Chapter 5. Designing a production imaging system	139
5.1 Design goal of the production imaging system	140
5.2 Capture system design	141
5.2.1 Document hierarchy	142
5.2.2 Capture processing tasks	143
5.2.3 Capture workflow	146
5.2.4 Capture design considerations	146
5.2.5 Discovering the capture process	163
5.3 Requirements gathering	164
5.3.1 Requirements for current capture or document processing environment	164
5.3.2 Processing location requirements	166
5.3.3 Document type requirements	167
5.3.4 Captured data requirements	168
5.3.5 Verification requirements	169
5.3.6 Export requirements	170
5.3.7 Volume and timing requirements	170
5.3.8 Administration requirements	171
5.4 Designing the capture for the auto claims scenario	171
5.4.1 Document hierarchy	171

5.4.2	Capture processing tasks	175
Chapter 6.	Implementing the capture solution	179
6.1	Configuring the Datacap application	180
6.1.1	Creating a blank application	180
6.1.2	Setting up the document, pages, and fields	189
6.1.3	Setting up the physical scan device	191
6.1.4	Creating a module in Taskmaster	192
6.1.5	Creating a job within the Taskmaster Client	193
6.1.6	Setting up the iScan task	195
6.1.7	Setting up the PageID task	198
6.1.8	Setting up the Rulerunner task	206
6.1.9	Testing the progress	217
6.2	Zones and fingerprints.	218
6.2.1	Setting up the zones	220
6.2.2	Setting up optical mark fields	220
6.2.3	Setting up an ICR field	223
6.2.4	Removing the Clean rule set.	225
6.2.5	Creating a rule set to read fields	226
6.2.6	Creating a rule set to validate fields	229
6.2.7	Validating captured field values against the database	231
6.2.8	Setting up routing	235
6.2.9	Testing scan validation and routing.	236
6.2.10	Setting up the verification panel	238
6.2.11	Setting up the Verify job	241
6.2.12	Setting up the export to the repository	246
6.3	Task profiles overview.	251
Chapter 7.	Adding a document type to an existing application	253
7.1	Adding VScan to a rule set	254
7.2	Adding a document with pages and fields.	254
7.3	Setting the PageID logic	256
7.4	Adding the Image Enhance feature to the pages	258
7.5	Configuring the CreateDocs rule set for the pages	258
7.6	Adding a full page OCR rule set to the pages	259
7.7	Setting the fingerprint for the Estimate_Invoice document	260
7.8	Obtaining field values in the Claim_Pg document by using the Locate() rule set.	261
7.8.1	Doc Title field	261
7.8.2	Pol_Number field.	262
7.8.3	Claim_Number field.	262
7.8.4	Vendor Number field	262
7.8.5	Vendor Name field	262

7.8.6 Ref ID field	262
7.8.7 Ref Date field	262
7.8.8 Ref Total field	263
7.9 Looking up vendor information from a database by using the Lookup() rule set	263
7.9.1 Looking up a vendor name	263
7.9.2 Looking up a vendor number	264
7.10 Validating data.	264
7.10.1 Page Level field.	265
7.10.2 Doc Title field	265
7.10.3 Pol_Number field.	265
7.10.4 Vendor Number field	265
7.10.5 Vendor Name field	265
7.10.6 Ref ID field	265
7.10.7 Ref Date field	266
7.11 Routing scanned pages.	266
7.12 Verify task with Batch Pilot	266
7.12.1 Creating a Verify panel	267
7.12.2 Determining which pages to display	267
7.13 Export task	268
Chapter 8. Implementing the business process component	269
8.1 FileNet Business Process Manager Tools	270
8.1.1 Steps.	270
8.1.2 Routes.	271
8.1.3 Maps	271
8.1.4 Content events	271
8.2 Running the Auto Claims process with the FileNet BPM Tools.	271
8.3 How the Auto Claim process works.	273
8.4 Business Process Manager step configuration	283
8.4.1 DBExecute step	283
8.4.2 Conditional step	286
8.4.3 Activity step.	287
Chapter 9. Best practices and recommendations	289
9.1 Basic form design and capture	290
9.1.1 Scanned document verification.	294
9.1.2 Measuring scan and capture process improvement	295
9.2 Best practices for application development.	295
9.2.1 Testing an application	295
9.2.2 Capturing data.	296
9.2.3 Smart parameters	297
9.2.4 Projects	297

9.2.5 Actions	297
9.2.6 Scripting	298
9.2.7 OMR field configuration	298
9.3 Production Imaging Edition implementation principles	300
Part 3. Advanced technologies	303
Chapter 10. Dynamic technologies	305
10.1 Introduction to dynamic technologies	306
10.2 PageID actions and techniques in dynamic applications	307
10.2.1 Page identification by barcode separator	310
10.2.2 Electronic input of documents	312
10.3 FlexID	313
10.4 DNA technology	314
10.4.1 Fingerprint matching in dynamic applications	315
10.4.2 Using the TemplateID	316
10.4.3 The offset	317
10.5 Sticky fingerprints	317
10.6 Managed recognition	321
10.7 CCO Merging	323
10.8 FPXML	324
10.9 Line item detection	325
10.9.1 Storing repeating structures	326
10.9.2 Zoning the detail structure	327
10.9.3 Capturing the detail structure	329
10.9.4 Filtering line items	330
10.10 Enhanced error messaging	332
10.11 Data localization actions	334
10.11.1 Actions that affect rules execution	334
10.11.2 Actions that affect the captured data	335
10.12 Intellocate	335
10.13 Flex technology	337
10.14 Conclusion	338
Chapter 11. Technical walkthrough Accounts Payable Capture	339
11.1 Introduction to Accounts Payable Capture	340
11.2 How IBM Taskmaster Accounts Payable Capture works	340
11.3 Jobs available in the workflow	341
11.4 When each task profile gets executed	343
11.5 A walkthrough of the task profiles	344
11.5.1 The VScan Task Profile	344
11.5.2 The Batch Profiler Task Profile	346
11.5.3 The Verification process	373
11.5.4 The Export Task Profile	376

Part 4. Application and performance	385
--	------------

Chapter 12. System scalability, availability, backup, and recovery	387
12.1 System scalability, performance, and availability	388
12.1.1 Typical Datacap installation	388
12.1.2 Scaling Rulerunner vertically (scale up)	389
12.1.3 Scaling Rulerunner horizontally (scale out).....	390
12.1.4 Scaling Rulerunner horizontally and vertically	392
12.1.5 Taskmaster Server scaling and redundancy.....	393
12.1.6 Scaling both Taskmaster Server and Rulerunner.....	396
12.1.7 Taskmaster Web scaling and redundancy	397
12.1.8 Scaling and redundancy for thick clients.....	398
12.1.9 Load balancing of tasks	399
12.1.10 Scaling databases.....	400
12.1.11 Network share drive	400
12.1.12 Scaling across geographies	401
12.2 Rulerunner.....	402
12.2.1 Single-threaded Rulerunner	402
12.2.2 Multithread Rulerunner	404
12.2.3 Load balancing Rulerunner.....	406
12.2.4 Race conditions.....	407
12.2.5 Running multithreading in VMware	407
12.2.6 Fingerprint Service	408
12.3 Configuring Rulerunner	409
12.3.1 The Datacap.xml and <project>.app files	409
12.3.2 Installing a single Rulerunner server.....	412
12.3.3 Installing an additional Rulerunner server.....	421
12.3.4 Configuring priorities and queuing	425
12.3.5 Installing Fingerprint Service.....	428
12.3.6 Using the Fingerprint Service	432
12.4 Adding additional Taskmaster Servers	434
12.4.1 Adding a failover Taskmaster Server	434
12.4.2 Load sharing between Taskmaster Servers	439
12.4.3 Overview of Rulerunner Server and Taskmaster Server	444
12.4.4 Scaling a thick client	445
12.4.5 Scaling a Taskmaster Web client	449
12.5 Backup and restore	449
12.5.1 Backing up and restoring Rulerunner machines	449
12.5.2 Backing up and restoring the Taskmaster Server.....	450
12.5.3 Backing up the database server	451
12.5.4 Backing up and restoring the Fingerprint server.....	451
12.5.5 Backing up and restoring the IIS web server	451
12.5.6 Backing up the file share.....	452

Chapter 13. Installation, migration, and application reuse	453
13.1 Installing the Datacap Taskmaster Capture software	454
13.1.1 Installing IBM Taskmaster Web client	456
13.1.2 Installing Taskmaster thick client	458
13.1.3 Installing the Taskmaster Server.	458
13.2 Migrating application from development to another environment	459
13.3 Reusing applications	463
13.3.1 Creating an application based on an existing one	463
13.3.2 Enhancing a new application with existing design elements.	465
Related publications	473
IBM Redbooks	473
Other publications	473
Online resources	474
Help from IBM	474

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.


Cognos®

FileNet®

Redbooks®

DB2®

IBM®

Redbooks (logo)  ®

developerWorks®

Lotus®

WebSphere®

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Organizations face many challenges in managing ever-increasing documents that they need to conduct their business. IBM® Production Imaging Edition V5.0 is the comprehensive product that combines imaging, capture, and automation to provide organizations the capabilities to process and manage high volumes of document imaging over their entire life cycle.

The advanced document capture capability of Production Imaging Edition is provided through the bundling of IBM Datacap Taskmaster Capture. As a key capture component, Datacap Taskmaster Capture V8.0 is also a new product in the IBM Enterprise Content Management (ECM) product family.

This IBM Redbooks® publication introduces Production Imaging Edition, its components, the system architecture, its functions, and its capabilities. It primarily focuses on Datacap Taskmaster Capture. It explains how Datacap Taskmaster Capture works, how to design a document image capture solution, and how to implement the solution using Datacap Studio. Datacap Studio is the development tool that designers use to create rules and rule sets, configure a document hierarchy and task profiles, and set up a verification panel for image verification.

This book provides insight into the advanced technologies that are used to create dynamic applications such as IBM Taskmaster Accounts Payable Capture. It includes an in-depth walkthrough of a dynamic application, IBM Taskmaster Accounts Payable Capture, which provides invaluable insight to designers in developing and customizing their applications.

In addition, this book includes information about high availability, scalability, performance, backup and recovery options for the document imaging solution. It highlights known best practices and recommendations for designing and implementing such a solution.

This book is intended for IT architects and professionals who are responsible for creating, improving, designing, and implementing document imaging solutions for their organizations.

Accounts Payable Capture versus APT: This book uses the name *Accounts Payable Capture* to refer to what was previously known as *APT*. Keep in mind that some windows in the application and window captures in the book might still use the name APT.

The team who wrote this book

This book was produced by a team of specialists from around the world working for the International Technical Support Organization (ITSO).

Jackie Zhu is an Enterprise Content Management Project Leader with the IBM ITSO in the US. Jackie joined IBM in 1996 and has more than 10 years of software development experience in accounting, image workflow processing, and digital media distribution. She is a Certified Solution Designer for IBM Content Manager. Currently, Jackie manages and leads the production of Redbooks publications about Enterprise Content Management. Jackie holds a Master of Science degree in Computer Science from the University of Southern California.

Ben Antin currently leads client technical professional consultants worldwide for IBM Capture and Document Imaging products. He has worked with IBM FileNet® Enterprise Content Management products for over 20 years. Ben specializes applying automating and optimizing document processes using ECM technology. He has applied ECM technology in the financial services, insurance, transportation, and government sectors. He holds a degree in Computer Science from the University of Iowa.

Moe Bryan is an Enterprise Content Management Senior Managing Consultant with IBM Software Group, Industry Solutions Development in the US. He works in the Industry Solutions group specializing in Datacap software and works with the software training group delivering Datacap implementation training. Moe joined IBM in 2010 as part of the Datacap acquisition. He has more than 15 years of experience in software development, management, and training users on the implementation of Datacap software.

Patrick Chesnot is an Enterprise Content Management Product Manager with IBM Software Group US. Patrick joined IBM in 2006 as part of the FileNet acquisition. He has more than 11 years of experience in FileNet ECM product management. He also has more than 22 years in the field of document management, imaging, and publishing systems, where he worked as a product specialist, systems consultant, and project manager. He holds an Electrical Engineering degree from the University of Rouen in Le Havre, France, and a Maitrise degree in Applied Foreign Language Studies (English/Spanish) from the Charles V Institute, University of Paris VII in France.

Ben Davies is an ECM Client Technical Professional in IBM Software Group in the UK. He has a variety of IT experience ranging from digital security to master data management to Enterprise Content Management. Ben holds a bachelor degree in Multimedia Computing from Liverpool John Moores University in the UK and joined IBM in 2007 as part of the IBM UK Graduate scheme.

Tom Stuart is a Worldwide Executive ECM Capture Specialist. He has directed the pre-sales engineering activities of Datacap, which provides streamlined solutions for expediting implementation of capture systems. He is responsible for developing easy-to-replicate techniques that help customers rapidly deploy complex data capture applications, most of which are integrated with other business applications. Tom is also closely involved with refining prototype solutions and is the application developer responsible for IBM Taskmaster Accounts Payable Capture and Taskmaster Flex. Tom's dual background in education and product development gives him unique perspective on the theory and practical application of data capture and data processing practices. He is widely quoted on technical issues and frequently addresses industry conferences.

Michael Vahland is an ECM Architect in IBM Software Group in Germany. Currently, Michael designs capture solutions. Michael joined IBM in 1997 and has more than 10 years of experience in professional services and technical presales in collaboration solutions and enterprise content management solutions. He is a Certified Solution Designer for IBM Content Manager and for IBM CommonStore E-mail Archiving and Discovery. He is also a Master Certified IT Specialist from the Open Group and Project Management Professional from PMI. Michael holds a Master of Science degree in Electrical Engineering from the Technical University of Munich in Germany.

Thanks to the following people for their contributions to this project:

Emma Jacobs
Jenifer Servais
ITSO US

Scott Blau
Noel Kropf
IBM Armonk, NY

Alexey Gorbunov
Claudia McGhee
Michael Schlachter
Charlie Wiecha
IBM Datacap Software Development Lab, Tarrytown, New York

Robert Ferin
David Jenness
Freddy Naime
Philip Page
Jay Taylor
Reginald Twigg
IBM Software Development Group, IBM US

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



Part 1

Production imaging

Production imaging is about helping organizations to more efficiently manage the documents that they need to conduct their operations and to meet their business goals. This part introduces the concept and need for production imaging solutions. It also introduces IBM Production Imaging Edition V5.0. This comprehensive product combines imaging, capture, and automation to provide organizations the capabilities to process and manage high volumes of document imaging over their entire life cycle.

This part includes the following chapters:

- ▶ Chapter 1, “Production imaging overview” on page 3
- ▶ Chapter 2, “System architecture” on page 35
- ▶ Chapter 3, “Production imaging functionality” on page 51



Production imaging overview

This chapter provides an overview of the production imaging space and an introduction to IBM Production Imaging Edition, which is a comprehensive product for managing the entire document imaging life cycle. This chapter focuses mainly on the advanced document capture component of the product, which is provided by IBM Datacap Taskmaster Capture (also referred to as *Taskmaster*).

This chapter includes the following sections:

- ▶ The business document problem and approach to its solution
- ▶ Introduction to Production Imaging Edition
- ▶ Examples of applications
- ▶ Conclusion

Important: This book focuses on Taskmaster because it is a new addition to the IBM Enterprise Content Management (ECM) product family. For detailed information about the other components and products included in Production Imaging Edition, see the following IBM Redbooks publications:

- ▶ *IBM FileNet P8 Platform and Architecture*, SG24-7667
- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547
- ▶ *Introducing IBM FileNet Business Process Manager*, SG24-7509

1.1 The business document problem and approach to its solution

Organizations today face many challenges in efficiently managing the documents they need to conduct their business. They have the perennial paper problem and the ever-increasing electronic document problem. One challenge is to control all types of media that are required to conduct businesses. Another challenge is to ensure continuity, consistency, and longevity across these media over the entire life cycle of the business processes. The approach to solving these problems is a production imaging solution.

Tip: If you are already aware of the problems associated to managing business documents and want to jump directly to their solution and learn more about the Production Imaging Edition product offering, skip this section and go to 1.2, “Introduction to Production Imaging Edition” on page 10.

1.1.1 Paper everywhere

The advent of such innovations as email, the web, instant messaging, and social media has resulted in more efficient communication and a significant reduction in the need to print information. Despite these technological advancements, many companies still rely heavily on paper to conduct a large part of their business.

Organizations use paper for the following common reasons among others:

- ▶ Historical. The organization has to deal with existing forms and documents and has no say in the decision to convert these hardcopy forms and documents into electronic formats. Sometimes, the number of existing forms and documents make it unrealistic to undergo such a conversion process.

- ▶ Legal. In some cases, legislation has not kept pace with new technology and still refers specifically to having records on paper (which holds probative value). In other cases, new laws take into account electronic media, but have not been challenged in court yet, which means that the safe option is to keep using paper.
- ▶ Practical. The low-tech portability of paper makes it the best option to reach customers anywhere, irrespective of affordability, access to infrastructure, technical dependencies, or administrative boundaries. For example, many companies that otherwise rely on electronic means of communication for marketing purposes still send correspondence by post to confirm important transactions that require the attention and signature of the customer. Similarly, many customers still prefer to send their official correspondence through registered mail to be tracked by a third party with the expectation that it will be deemed an official record.
- ▶ Technical. Paper is the ultimate “systems integration technology,” a variant of the previous reasons. It is common to hear that the best way to communicate with another department in the same company or administration is to print the information and send it out. This is especially true in organizations where records keeping of inbound and outbound communication is available for mail, but not across incompatible business or ECM systems.

1.1.2 Business challenges posed by paper

If many businesses cannot do without paper altogether, dealing with paper still poses several challenges, as explained in this section.

For example, paper is expensive to store for the long term and to preserve under optimal conditions for business, legal, disaster (flood and fire), security, and safety reasons. Consider the costs incurred in maintaining shelf space, physical filing systems, and cabinets. Such costs also include robotics, temperature, and hygrometric control; fire and flood protection; and periodically verifying that the contents do not deteriorate over time or under heavy usage.

Paper is inefficient, time-consuming, inflexible, and expensive to manipulate in the course of conducting day-to-day business. For example, a home loan or credit card department might receive hundreds or thousands of applications from customers each day. These hard copy documents must be logged into their systems. In addition, the information on all these applications must be entered into the electronic loan processing systems of the department. Manually handling these applications are inefficient, time-consuming, and prone to human error.

As another example, for a particular document, you might have to go to the file cabinet, control who has clearance to access the document, track who has had physical access to it, and flag it as checked out. You might also have to make

copies of the document to share with people working the case or project and write comments on the document to pass on to the next person in the process. You might also have to do all the other tasks that you have to do with such information in a business process.

In a call center, it might be inconceivable to operate this way. Still for many organizations, productivity and access speed were not as critical initially. However, they now find themselves in a situation where they can no longer afford to operate in this way as competitive pressures and volumes increase.

Physical documents are exposed to the human factor. For example, they can be more easily lost, misfiled, or misclassified and never recovered. Their contents can be misread and entered with errors. Contents can also be discovered or used beyond their authorized purpose. The consequences can be expensive for a business.

More organizations are concerned with compliance. They want to ensure that they preserve the right documents while discarding documents that are no longer needed for the business. They also want to ensure that they purge documents as required after a certain period, as mandated by law, such as in some European countries. Although records management systems have long been available to manage physical records, it is increasingly challenging to manage records efficiently as the volumes of records increase.

The information on physical documents is locked on paper, which does not lend itself to automation and verification. As a result, transcription errors spread, and original mistakes go undetected. Spelling variations and typographical errors in names and nomenclatures cannot be verified and corrected until late in the business process, which is more expensive to fix. Also the business process cannot be automatically driven by the data. Therefore, dedicated expensive labor is required to do the low-value activity of routing documents to the next person in the process.

1.1.3 Business challenges posed by electronic documents

In many cases, organizations now use electronic documents in their business processes. For some of these processes, especially those processes that include external cycles with customers and business partners, it is still difficult to implement a solution that relies on one type of electronic media from end to end.

For example, it is now common to exchange electronic documents in at least part of the processing of a mortgage loan, from application to closing. However, especially in the case of a brokerage company, the organization is bound by legal requirements for paper. Also, often it cannot impose too many constraints on its customers and the other parties involved, such as lenders and assessors.

Therefore, the organization must accommodate many situations. It continues to accept the receipt of paper documents, which leads to the need to handle multiple types of media over the course of working on a mortgage loan.

The loan application goes through the following process:

- ▶ The initial loan application form on the web typically feeds into a business database. The form is used to generate an initial loan offer that is sent to a customer on paper or electronically as a PDF document.
- ▶ Preliminary contacts between the broker and the customer generate several email messages with attachments back and forth to each other.
- ▶ Many forms (disclosures, authorizations, estimates, and so on), typically in PDF are also exchanged. Signed copies must be captured, which leads to faxing or scanning documents that were generated electronically at first and then sent by email.
- ▶ The final loan settlement statement is signed. Each page is initialed by the customer at the formal closing meeting with the escrow officer. Then each page is scanned in and stored as the main document of record.

As you can see this process, although faster than if it were conducted entirely through paper and post, can be inefficient. It causes discontinuities in the media (from electronic text to paper to electronic image) and communication (email messages or faxes going in and out) over the life cycle of a given document. This challenge makes it difficult to ensure data consistency, control, and reconciliation with the business process and the transactions of the line of business (LOB) systems in-house.

Also, typically these documents must be archived over the long term in a final form. This form must ensure preservation of the formatting and contents across multiple generations of technologies without the need to use the original application that created it.

1.1.4 Solving the problem with production imaging

These challenges align precisely with the business objectives that many organizations hope to achieve with ECM systems, including production imaging. Figure 1-1 on page 8 shows a recent study conducted by the Association for Information and Image Management (AIIM)¹ on business objectives in ECM systems.

¹ AIIM State Of The ECM Industry Report, 2011, 650 responses

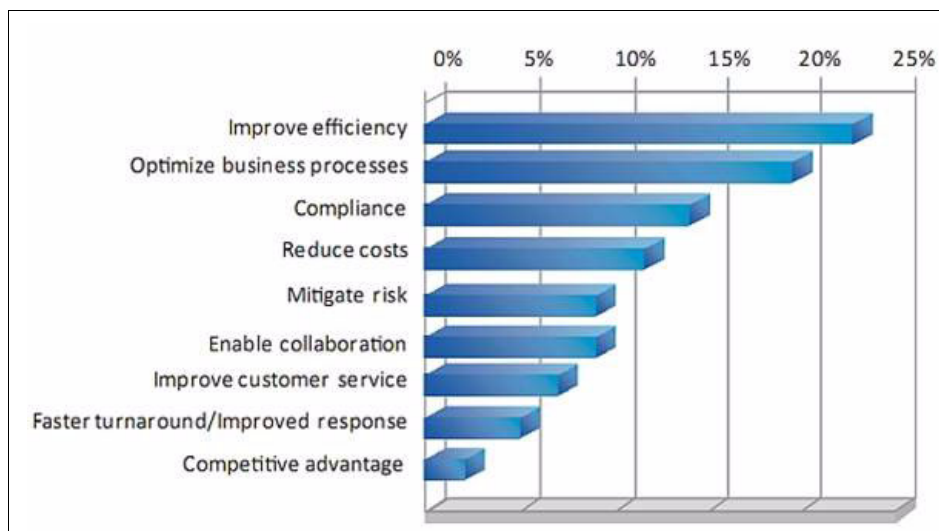


Figure 1-1 Business objectives in ECM systems

Production imaging is about helping organizations to manage more efficiently the documents they need to conduct their operations and meet their business goals. It is a set of capabilities that rely on technologies to achieve the following objectives:

- ▶ Reduce or eliminate the external paper cycle as much as possible to limit or remove the need to print, copy, store, and manipulate paper, and ultimately reduce cost.
- ▶ Capture customer input straight at the source in virtual forms (such as self-service portals and electronic documents) to limit errors, reduce delays and unnecessary steps, optimize processes, and reduce labor.
- ▶ Integrate document processing systems and repositories with LOB systems to reduce the need for paper, to reduce errors and normalize data, and to minimize labor.
- ▶ Digitize paper documents to feed into internal business processes as early as possible. This objective allows sharing and optimizes the processing of documents downstream, while meeting the scalability and deployment requirements that are typical of large organizations.
- ▶ Automate scanning, classification, separation, and data extraction of paper documents to minimize the added labor cost of scanning operations.
- ▶ Automate the import and conversion of electronic documents and email messages by using the same infrastructure as the one used for paper scanning to benefit from the same data normalization and validation rules.

- ▶ Verify and normalize data against business rules and databases and reduce errors. These types of issues are always more expensive to fix after they spread downstream.
- ▶ Provide flexibility to distribute image-processing operations (from local, to departmental, to central) to use available resources or easily shift them (such as when dispatching resources in disaster areas for the insurance industry).
- ▶ Store, classify or file, and secure content to avoid losses, provide protection, enforce compliance, manage records, and enable sharing.
- ▶ Provide context intelligence to drive automation of the business process and reduce delays, streamlining the allocation of work.
- ▶ Automatically deliver data and documents to users in a context that is the most relevant to the business process, using business process management technology. In addition, balance work loads, track work items, and gather statistics to help manage and tune up the business processes and increase efficiency.

Many examples are available of practical goals that some of our customers have attained by implementing production imaging solutions.

For example, a university was able to dramatically cut costs and improve productivity and customer satisfaction. It freed up much of the storage space that was previously occupied by paper archives and streamlined the flow of information in the organization. In making these changes, the university saved significant time for their staff, enabling them to focus more on their priorities. They also improved customer service by providing faster access to customers' correspondence.

In another example, a state tax department was able to improve its operations while providing jobs to residents of the state. It reduced the labor-intensive work that was needed in scan preparation and fixing data entry errors. It was able to double data entry productivity and achieve accurate reporting on operations and resource utilization. The department can now also access easily any tax return in the system. Of equal importance, the tax department can now use local labor and stimulate low income areas of the state by offering remote document processing positions.

As yet another example, a global logistics company reduced penalties and fines by processing shipping documents faster and meeting service level agreements. At the same time the company improved compliance with regulations and reduced overall processing costs and the number full-time employees.

As a final example, a healthcare insurance company achieved an average of 50% reduction in personnel required to process claims. The company also

achieved faster turnaround times, a reduction in duplicate claims submitted, and increased accuracy in claim processing.

By implementing production imaging solutions, these organizations were able to solve many of their business document problems.

1.2 Introduction to Production Imaging Edition

IBM Production Imaging Edition is a comprehensive product that provides the capabilities to process and manage high volumes of image documents over their entire life cycle, throughout thousands of users in the enterprise. For the first time, a solution is available that provides *extensive production imaging capabilities in a single, pre-integrated offering, from a single vendor*.

Production Imaging Edition includes software that enables you to perform the following tasks:

- ▶ Capture documents and turn them into digital images, extract information from them, and insert images and metadata into a repository.
- ▶ Organize, secure, and manage document images.
- ▶ Create and maintain business processes and automatically route document images based on content, context, and user input.
- ▶ View, annotate, and redact document images from anywhere in the enterprise.

Production Imaging Edition and its add-on components address various imaging use cases, including integrations with business applications and customer information databases. You can use any type of business document, anywhere in the organization, from a central location, remote office, multifunction printer, fax machine, email system, or from the desktop of someone on the go. Then with no or minimal manual operations, you can enter it in the system so that it can be immediately processed by business users with maximum efficiency.

With Production Imaging Edition, efficiency is derived from extracting pertinent data from the source paper document to automatically classify, index, organize, and route the document to the appropriate people or to an automated processor. The data is extracted by using Optical Character Recognition (OCR) or Intelligent Character Recognition (ICR), barcodes, or other image technologies. When the source is electronic documents, such as email messages and attachments, add-on components automatically extract and convert the individual attachments to a format that is suitable for long-term preservation.

Production Imaging Edition also helps to ensure that the data is accurate (true to the original) and correctly formatted and normalized against your vendor database, client database, or purchase order system. It also helps to ensure that the data can be used reliably in the business process. This capability applies to imported electronic documents, providing data consistency and validation across all the types of media used for your business documents.

Production Imaging Edition provides business process automation and optimization tools to route the documents reliably and present them to users in a business context. With relevant supporting business data, processes can be automated against business rules, and when users are involved, they can make their business decisions quicker.

The ability to use a single viewing tool to display the various types of documents and formats that are typically in a dossier or case you are working on is an important feature of the Production Imaging Edition offering that improves user productivity. You do not have to work with the various office applications that are needed for each format. With this viewer, you can also annotate reliably documents for the next person and redact information that is not supposed to be shared outside your workgroup. Furthermore, with the viewer, users can manipulate large documents efficiently without having to wait for pages to display.

1.2.1 Components of the Production Imaging Edition offering

The Production Imaging Edition offering includes the following components in one solution:

IBM Datacap Taskmaster Capture

An advanced document capture software.

IBM FileNet Content Manager

ECM software.

IBM FileNet Business Process Manager (BPM) Tools

The business process management tools that are included in FileNet Content Manager.

IBM Production Imaging Edition Viewer

A viewing, annotation, and redaction software.

Bulk Importer Tool (BIT)

A high-speed document ingestion software

IBM Datacap Taskmaster Capture

IBM Datacap Taskmaster Capture (Taskmaster) handles production-level digitization, data extraction, verification, indexing, and commitment of documents to back-end systems. It includes the following components:

- ▶ Server components to manage and serve the images that have been scanned or imported
- ▶ Thick and thin clients to handle user-attended tasks such as scanning, indexing, verification of metadata, and administration of users and security
- ▶ Rulerunner, which is a rule engine to execute unattended capture operations such as image cleanup, data extraction, lookup, redaction, and export of contents and metadata to back-end systems
- ▶ Configuration, reporting, and system monitoring tools

IBM FileNet Content Manager

IBM FileNet Content Manager constitutes the enterprise content repository of the production imaging solution. FileNet Content Manager stores, organizes, and circulates documents among users under the control of a business process (workflow). It includes the following components:

- ▶ Content Engine to provide the content repository and content management capabilities
- ▶ Process Engine to execute workflows
- ▶ IBM FileNet Workplace XT client to provide the interface for users to manage content and process work items that are circulated through business processes

IBM FileNet Business Process Manager Tools

IBM FileNet BPM Tools help to design, manage, and monitor business processes. This component includes the following tools:

- ▶ IBM FileNet Business Process Designer and Connector for Microsoft Visio, which enable the creation and update of business process definitions
- ▶ IBM FileNet Business Process Tracker, which enables the monitoring of business process instances
- ▶ IBM FileNet Business Process Simulator, which enables the simulation and validation of a process before placing it into production
- ▶ IBM FileNet Case Analyzer, which enables the gathering of business process statistics for analysis
- ▶ The ECM Widgets and the IBM Lotus® mashups, which enable customers to design their own paneled views

These tools are available through IBM FileNet Content Manager.

IBM Production Imaging Edition Viewer

The Production Imaging Edition Viewer provides extended capabilities to view, annotate, and redact PDF documents and other types of electronic documents without needing to have the native applications installed. Production Imaging Edition Viewer is based on Daeja ViewONE Pro.

Bulk Importer Tool

The Bulk Importer Tool (BIT) provides high-speed ingestion of documents into the content repository. The tool is delivered on the FileNet Content Manager distribution media.

Tool delivery and use: At the time of writing this book, the BIT is delivered on the FileNet Content Manager media but is licensed for use through Production Imaging Edition.

1.2.2 Taskmaster base and add-on components

In addition to the Taskmaster components included in Production Imaging Edition, Taskmaster offers add-on components to expand functionality and flexibility of your production image solution. These add-on components can be used through separate licenses of the Taskmaster software.

Taskmaster components and functions offered through Production Imaging Edition

Production Imaging Edition provides the functionality and components listed in Table 1-1 that are delivered in the Taskmaster base package. They include a set of sample applications that can be cloned and used as a base to configure your own application, providing you a head start in the process.

Table 1-1 Taskmaster base components

Component name	Functionality
Taskmaster Server	Manages and serves documents and executes Taskmaster workflow
Taskmaster Rulerunner	Executes processing rules on documents
Taskmaster Web	Provides a web interface for user-attended tasks and administration
Taskmaster Client (thick)	Provides a thick client interface for initiating user-attended tasks and administration

Component name	Functionality
Batch Pilot and Dot Edit	Configures graphical user interfaces (GUIs) for thick clients
Datacap Studio	Configures and tests applications
Flex Manager	Configures data and document types for the Flex Capture sample application
Application Manager	Maintains a registry of applications
RV2	Reports on runtime activity
NENU	Monitors operations and automates recurring tasks
Flex Capture sample application	Processes documents with variable layouts
Survey sample application	Processes surveys
1040EZ sample application	Processes US tax returns
Express sample application	Scans and indexes documents remotely on the web

Add-on components of Taskmaster

The add-on components address additional connectivity and scalability needs. They can also provide an imaging solution as close to turnkey as possible in specific functional areas. These components require additional licensing agreements.

Table 1-2 lists the functionality available through the add-on components of Taskmaster.

Table 1-2 Taskmaster components subject to additional licensing

Component name	Functionality
Rulerunner Enterprise	Multithreading and enhanced Fingerprint services
Connector for Email and Electronic Documents	Imports email attachments from Exchange and Internet Message Access Protocol (IMAP) mail servers and converts electronic documents
Connector for OpenText LiveLink	Exports document images and data to an OpenText Livelink repository
Connector for EMC Documentum	Exports document images and data to the EMC Documentum repository

Component name	Functionality
Connector for Microsoft SharePoint	Exports document images and data to Microsoft SharePoint
Connector for Rightfax	Imports fax images from an OpenText RightFax server
Accounts Payable Capture	Solution to process invoices and similar documents
Medical Claims Capture	Solution to process US healthcare claim and explanation of benefits forms

Getting the add-on components: Taskmaster includes all base and add-on components on its own distribution media. However, to use the add-on components, you must acquire the necessary additional licensing from IBM.

1.2.3 The production imaging process

To understand the nature of the components included in Production Imaging Edition and how they fit together to meet the requirements of imaging use cases, you must understand the typical life cycle of a document in an imaging system. This life cycle usually consists of two phases: the precommittal process and the postcommittal process.

Precommittal process

From a document perspective, the precommittal process deals with the steps that take place *before* the creation of the document in an ECM repository. This process is handled by the Datacap Taskmaster Capture software in Production Imaging Edition.

At this stage, the document does not exist in the enterprise repository yet. Hence, its processing by business users, using the business process management infrastructure, has not started. The precommittal process is driven by the Taskmaster workflow and generally includes the following tasks:

- ▶ Scanning
- ▶ Image processing
- ▶ Separation and classification of images
- ▶ Data extraction
- ▶ Data validation
- ▶ Preparation for indexing

In the precommittal phase, the documents are *collections (or batches) of independent pages*. They cannot be manipulated and used as documents in the conventional sense by business users yet.

The pages are raster images straight from the scanner, or alternatively, native electronic documents, if they are imported. Their format is not necessarily the one you want to store in the ECM repository. This format is typically the file format that is most appropriate for image processing, for OCR or ICR and barcode. In certain situations, you might want to convert the format to a different format for long-term preservation or other needs.

During the precommittal phase, and depending on the requirements of your imaging operations and volumes, you typically dedicate Taskmaster users to this process for efficiency reasons. (These users are *not* technically ECM users.) They handle document scanning, data entry, verification, and processing exceptions.

Overall, the precommittal process must be as short as possible, occurring in minutes or hours at the most. The short time frame is necessary because the business document, as known by the ECM system, has not been created yet. Therefore, it has not been officially received and has no date of record, apart from the possible date stamp that might be used to endorse the document at scan time. From a business perspective, it is not optimal if the precommittal process takes too long, especially if it means delaying revenue. The longer it takes for the documents to progress through the precommittal phase, the longer the delay is to start the associated business processes.

Postcommittal process

The second phase, known as the *postcommittal process*, starts with the creation of the document in the ECM back-end system, FileNet Content Manager. It spans the entire life cycle of the document up to its disposal. During this phase, all the functionality of FileNet Content Manager, including content and business process management, can be used to the fullest extent.

At this stage, the document is now a conventional ECM document, and it consists of one to many pages, with searchable metadata that have been extracted from the pre-committal processing phase. The document is classified based on the document types that are defined in the content repository. It is typically filed in a folder structure that meets the business requirements. Documents can be searched for or randomly accessed by browsing. More typically in an imaging application, the documents are distributed to business users through work items that are circulated through a workflow.

Documents can be associated with a FileNet Content Manager workflow in one of two ways:

- ▶ The document initiates a new workflow instance upon entering the FileNet Content Manager system. The document gets attached to the new workflow.
- ▶ The document reconciles automatically with an already running workflow instance when specific conditions are met.

In addition to the document being attached to a workflow instance, the properties of the document can be transferred to the FileNet workflow data fields so that they can be used to automate the business processing logic. The document properties include the data that was captured in the precommittal phase by Taskmaster. For example, these properties can be used to evaluate routing conditions or the completeness of the data gathered or to interact with a rating or business rule engine.

In some cases, depending on business requirements, the document can be converted automatically to another format by using the FileNet Rendition Engine. It can be viewed, annotated, or redacted while it is being circulated and processed by users. Also functionality can be extended by adding licensing for IBM Case Manager when additional flexibility and collaboration are needed for processing collections of data and documents that belong together and require coordination.

The ECM Administrator declares users who are involved in the postcommittal business process as ECM users. The postcommittal process expands over the entire life cycle of the documents, possibly over years, until they are purged from the system based on business requirements, rules, and regulations.

Figure 1-2 shows an example of a production imaging process.

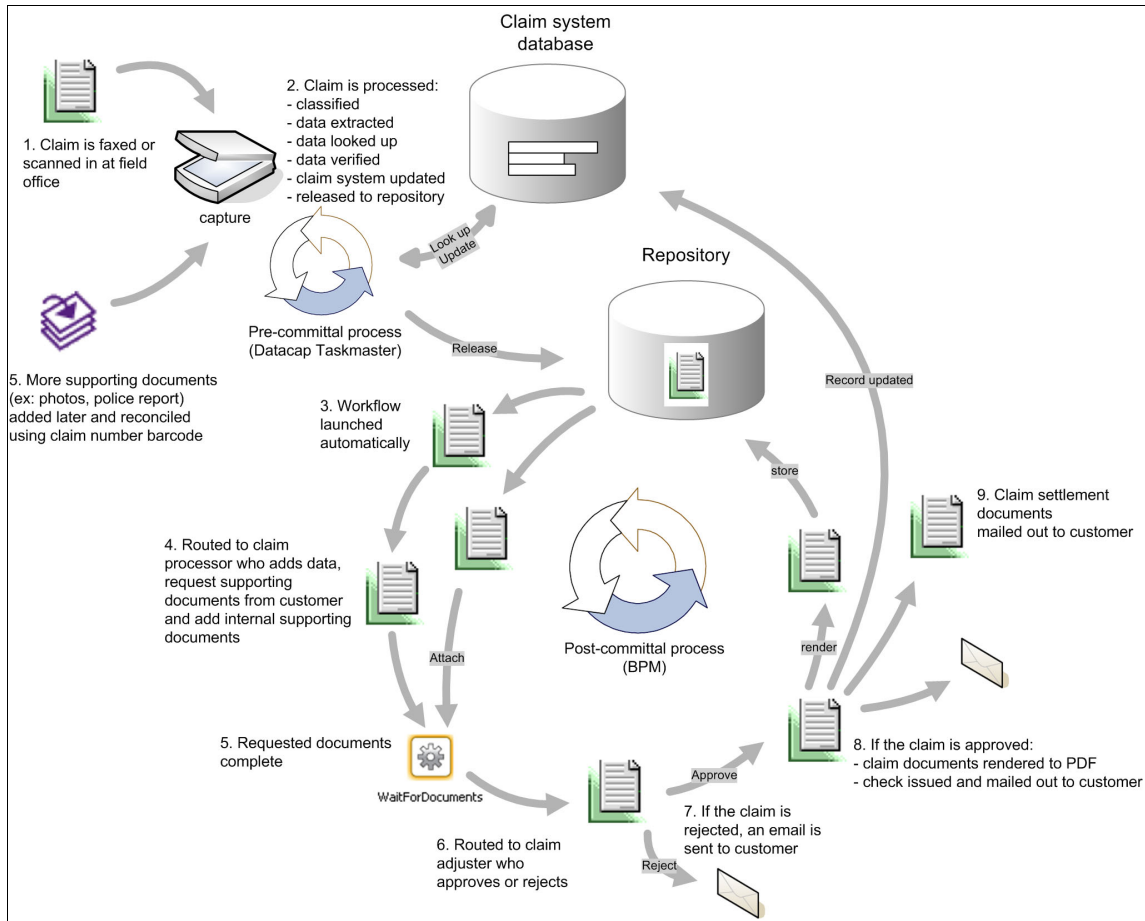


Figure 1-2 Example of an overall production imaging process

New possibilities that blur the boundaries

The architecture of Taskmaster can also enable use cases where documents can be processed less linearly than as described previously.

For example, as shown in Figure 1-3, you can ingest a batch quickly in FileNet Content Manager with minimal data recognition and processing to make the documents available to the ECM users. At the same time, you can continue to execute tasks in the Taskmaster workflow, such as some user-attended operations, or wait for conditions to complete. Upon completion of the outstanding tasks, Taskmaster can automatically update the metadata of the documents in FileNet Content Manager.

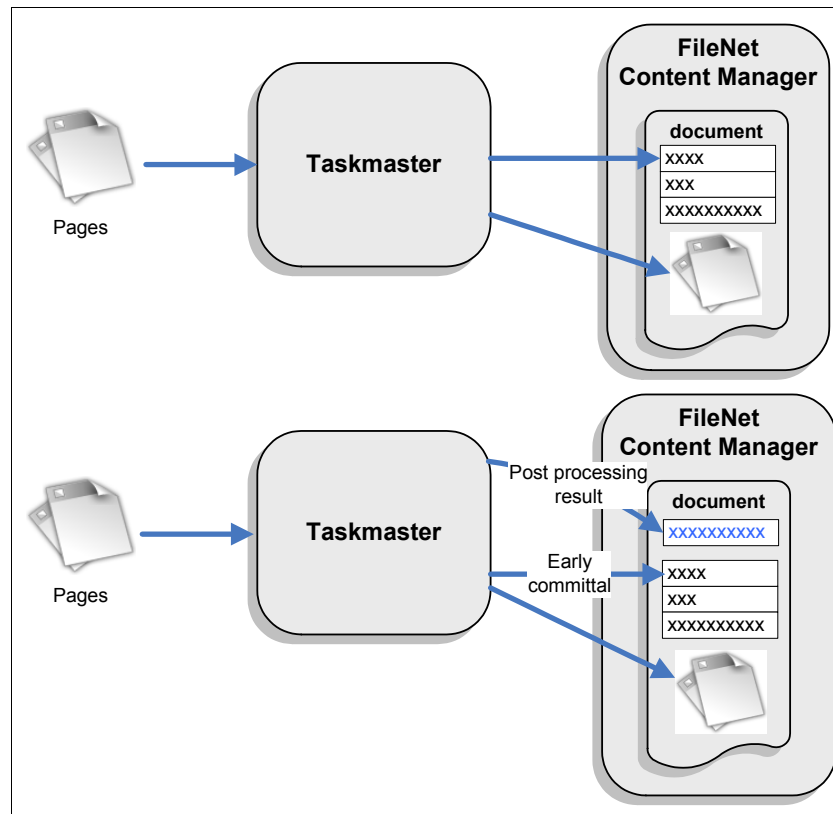


Figure 1-3 Nonlinear processing showing postcommittal indexing by Taskmaster

In another case, as shown in Figure 1-4, documents can be ingested in FileNet Content Manager first, and then a workflow can be started. Then, at specific steps in that workflow, Rulerunner actions can be started to validate or perform math functions on data added to the BPM work items. Alternatively, these actions can look up information in a business database to normalize the data as the workflow progresses.

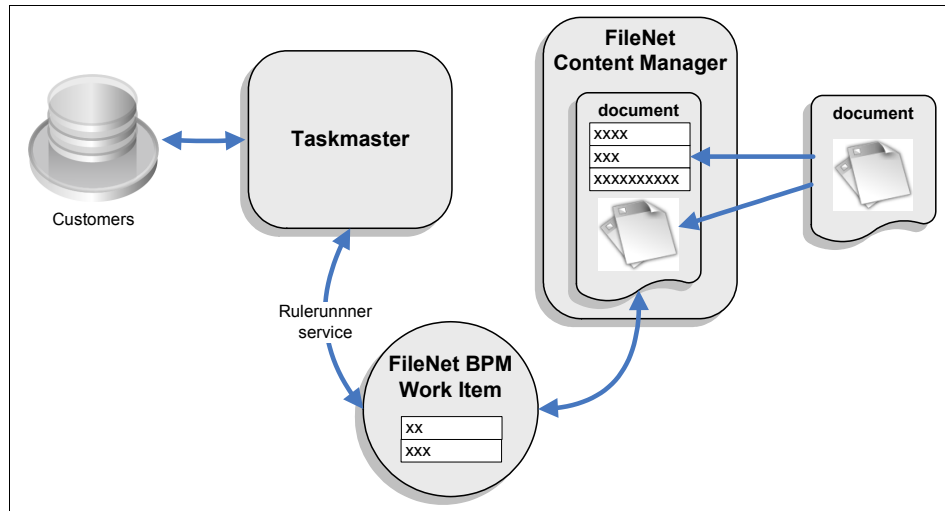


Figure 1-4 Nonlinear processing showing the usage of Rulerunner

In another example as shown in Figure 1-5, you can have a process in which some documents are ingested in FileNet Content Manager, starting a workflow. Again at specific steps, the Taskmaster application can be called in to perform specific tasks, such as starting a scan or importing a job to attach additional supporting documents to a case.

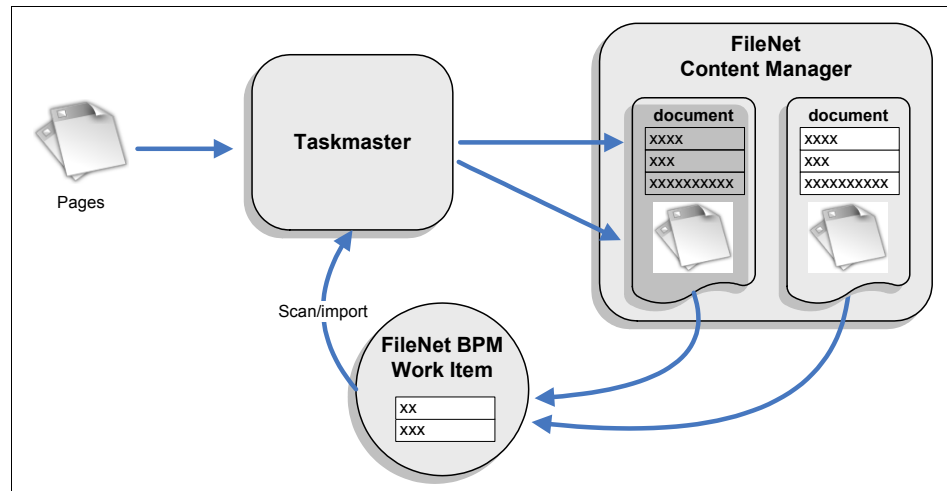


Figure 1-5 Nonlinear processing showing the usage of Taskmaster scan or import

To implement a solution for these use cases, customization and modification are required.

1.2.4 Focus on Taskmaster

The key components that make any production imaging solution possible are the components that provide document digitization. However, the components that automate data extraction, classification, and separation are of equal critical importance to deliver an efficient and cost reduction solution. These activities are otherwise labor-intensive and represent major running expenses that grow as a proportion of the document volumes and that limit returns on investment. In this regard, Taskmaster, with its advanced automation capabilities, is a major asset and a differentiator of the Production Imaging Edition offering.

Taskmaster is also the latest new product in the IBM ECM product family. For this reason, this book focuses specifically on the functionality of Taskmaster. The other main components of the solution, such as FileNet Content Manager and BPM tools, are covered at a much higher level.

For more detailed information about FileNet Content Manager and BPM tools, see the following IBM Redbooks publications:

- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547
- ▶ *IBM FileNet P8 Platform and Architecture*, SG24-7667
- ▶ *Introducing IBM FileNet Business Process Manager*, SG24-7509

1.3 Examples of applications

This section reviews the types of applications where you can take advantage of Taskmaster and production imaging technologies.

Taskmaster can be used in many scenarios, from simple to complex, from handling simple correspondence documents with no structure, to complex forms with intricate layouts, such as those found in the healthcare industry. The more complex the data structure and the density are, the more specialized and complex the application is, and the more resource-intensive the processing is.

Central to the implementation of any Taskmaster solution is the concept of an *application*. An application unites a set of Taskmaster capabilities with the aim of solving a specific business need. By combining the set of functions that ready for immediate use, Taskmaster can address various use cases. Its sample and add-on applications, which are geared at addressing specific business situations, are not necessarily confined to these domains. For example, consider the functionality that is implemented in Taskmaster Accounts Payable Capture to process invoices with line items. You can use this functionality in other domains where there is a need to process documents with similar characteristics, such as in transportation and logistics, with shipping manifests, or in manufacturing with nomenclatures.

However, you must keep in perspective that the reach of the applications extends beyond the pure document capture aspect. Combining them with the flexibility offered by the web deployment options of Taskmaster and the repository and business process management infrastructure of FileNet Content Manager provides a true enterprise-wide imaging solution that can use resources anywhere in the organization.

1.3.1 Cross industry: Automated forms processing

You can achieve significant savings by automating data entry with forms processing software. Taskmaster reduces or eliminates expensive typing of data and delivers data seamlessly to your business applications quickly, accurately, and more cost effectively than manual methods.

Taskmaster applications apply various technologies to locate, extract, and validate the accuracy of data from several forms, including health claims and tax returns. This functionality also applies to unstructured and semi-structured forms, such as invoices, shipping bills, and explanations of benefit.

Taskmaster can capture handprint, machine print, check boxes, and barcodes, including combinations on the same document. The dynamic reusable rules of Taskmaster deliver a high level of flexibility over every aspect of the capture process. They can be used across multiple types of forms to normalize data and ensure consistency. You can choose from hundreds of prebuilt rules in the action libraries, modify them, or create new ones by using Datacap Studio, which is the intuitive point-and-click configuration environment of Taskmaster. You can build complex forms processing workflows, without expensive programming, and test and implement them in a short time.

1.3.2 Cross industry: Distributed capture

Great savings in shipping costs can be achieved if you are able to scan documents at the point of origin and send them electronically. This same concept applies if your operators in charge of verification can work from anywhere with a workstation and a browser, whether at home or in a low cost area. Cost savings, speedier input, and more IT flexibility for your organization are among the benefits of distributed capture.

Taskmaster Web is a thin-client capture software that enables browser-based scanning and verification or indexing. You only need to download ActiveX components for remote scanning and indexing, greatly reducing implementation and administration headaches. Taskmaster Web provides a seamless integration to your back-end business applications and image repositories. It can be easily assimilated into your existing environment, reducing expenses without disruption.

One key to the success of a distributed capture solution is comprehensive oversight by an administrator. To this end, Taskmaster Web provides administrative tools to monitor and report on all work done remotely. A user logs in to Taskmaster Web with a password and starts working. All user permissions and privileges are centrally controlled by the administrator.

1.3.3 Cross industry: General business documents processing

Virtually any business can benefit from the automated classification and data capture technology of Taskmaster to reduce costs and improve document processing time for various back-office documents. These documents include the following types among others:

- ▶ Inbound sales orders and subscriptions
- ▶ General business correspondence
- ▶ Human resources documents, such as job applications, resumes, beneficiaries statement, withholding forms, reports, and contracts
- ▶ Marketing documents, including free-form documents, data sheets, product descriptions, press releases, announcements, and white papers

One way to achieve results quickly when processing various document types is to use the flexible capture capabilities of Taskmaster. With minimal configuration required, you can configure the document types for your structured and semistructured information. Then, you can let Taskmaster automatically locate data and assist you with adding new document types with a few clicks as you go.

Taskmaster relies on a unique feature called *fingerprinting* to identify incoming documents based on their layout and match them to known document types. After a type of document is recognized, the data is located and extracted automatically. In most cases, no operator needs to get involved.

However, in some cases, the document is of an unexpected type, or there is so much variability that its type cannot be determined with high confidence. In such cases, the document must be processed manually by an operator who determines the type of document that is being processed. The operator visually locates and points in the image to the key data that defines the document type. Examples of such data include an invoice number, purchase order, or vendor name, as in the case of an invoice.

From this manual processing, Taskmaster can record the recognized data and its location in the image. Then it uses this information to automatically recognize and process similar documents in the future. Over time, the exceptions become less frequent. You do not need to go through an extensive, time-consuming training and setup phase to process your documents. Also you can start document capture quickly and improve the process as you go, which is called *flexible capture*.

The Flex sample application, together with the Flex Manager configuration tool, provide all you need to set up your flexible capture operations quickly. The verification task shown in Figure 1-6 on page 25 has a simple GUI layout. This

layout is populated automatically with the appropriate fields and matching image snippets based the type of document that has been recognized.

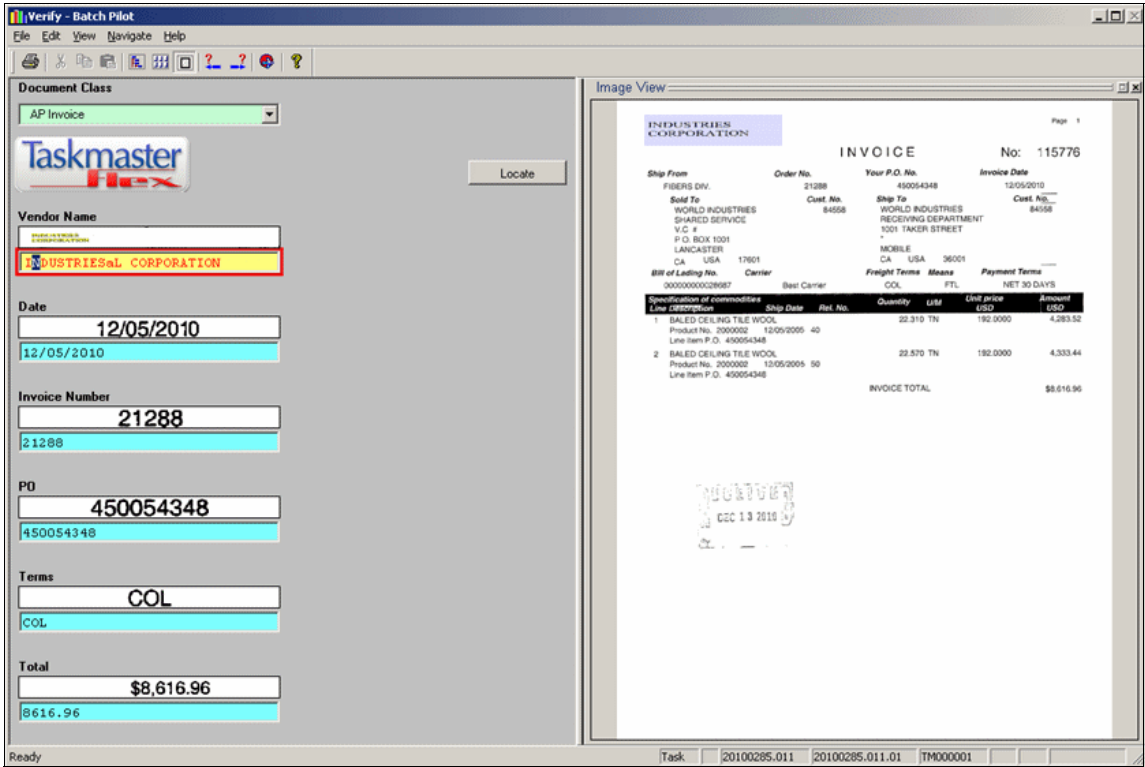


Figure 1-6 Sample application showing flexible capture

For text-intensive, free-form documents, Taskmaster uses the Wordfire connector and IBM Classification Module to classify and separate documents automatically. It eliminates the need for tedious manual prescan preparation, allowing you to process batches that contain multiple types of documents.

1.3.4 Cross industry: Accounts payable

Organizations that process invoices manually or scan without the help of optical recognition technology can significantly increase efficiency and accuracy through recognition-assisted automation.

Taskmaster Accounts Payable Capture is a solution specifically designed for processing invoices. The application automatically extracts all the important information from an invoice, including line items, and delivers it to your business application with no manual data entry.

Taskmaster Accounts Payable Capture ships with special invoice-centric functionality. It is designed for the unique requirements of Accounts Payable. It includes the ability to apply global rules to all or specific types of invoices, run multiple rules per field, automatically find fields and line items over multiple pages, reconcile purchase orders, and attach vendor numbers. Figure 1-7 shows the user interface of Taskmaster Accounts Payable Capture.

The screenshot displays the Taskmaster Accounts Payable Capture application interface. The main window is titled "Payables | Demo.Verify on batch 20110167.012 - Taskmaster DotEdit". It features a menu bar with options: Navigate, Edit, View, Snippet, Image, Help, Run Rules. Below the menu bar is a toolbar with icons for image view, search, and other functions. The interface is divided into several sections:

- Invoice View:** Displays the scanned invoice for "Protection Company". The invoice includes customer information (Name, Address, City, State, ZIP, Phone), invoice details (Invoice No. 36149, Date 02/15/10, Order No. 118839, PO # 013125, Terms 2/10Net30), and a line item table.

Qty	Item	Description	Unit Price	TOTAL
1	1000-NM	Base Charge	2100.00	2100.00
1	1100-NM	Mileage	3.00	3.00
1	3200-NM	3" Remote Valve	500.00	500.00
1	13001-NM	Additional Man	400.00	400.00
1	13001-NM	Discount	-1100.00	-1100.00
			SubTotal	2050.00
			Shipping	107.63
			Tax 5.28%	107.63
			TOTAL	2157.63
- Find Details / Calculate Blank:** A section for entering invoice details.

Vendor	Protection Company	
Vendor_Number	55028	Invoice_Type
TS63689	55028	PO
Invoice_Number	02/15/10	PO_Number
36149	02/15/2010	013125
		013125
Tax	Shipping	Invoice_Total
107.63		2157.63
107.63		2157.63
- Lineitem0 (1/5):** A table showing the line items from the invoice.

ItemID	ItemDesc	Qty	Price	LineTotal
1000-NM	Base Charge	1	2100.00	2100.00
1000-NM	Base Charge	1	2100.00	2100.00
1100-NM	Mileage	50	3.00	150.00
1100-NM	Mileage	50	3.00	150.00
3200-NM	3" Remote Va	1	500.00	500.00
3200-NM	3" Remote	1	500.00	500.00
13001-NM	Additional Man	1	400.00	400.00
13001-NM	Additional Man	1	400.00	400.00
- Batch View:** A list of batches on the right side of the interface.
 - 20110167.012.01 Invoice
 - TM000001 Main_Page
 - 20110167.012.02 Invoice
 - 20110167.012.03 Invoice
 - 20110167.012.04 Invoice
 - 20110167.012.05 Invoice
 - 20110167.012.06 Separator
 - 20110167.012.07 Invoice

The status bar at the bottom indicates the current batch is 20110167.012 and the document is 20110167.012.01.1(7) Image TM000001.1(1).

Figure 1-7 Accounts Payable Capture add-on application for capturing invoices

The Taskmaster Accounts Payable Capture add-on application includes the following features:

- Processes new invoices dynamically.
- Automatically attaches the vendor ID number to known invoices.
- Supports multipage invoices and line item capture.
- Looks up data and checks math to ensure accuracy.
- Provides plenty of flexibility to configure the rules to manage your invoice.
- Formats data to feed into Accounts Payable systems.
- Provides for easy purchase order reconciliation at data entry.
- Enables you to notify the system administrator by email that a new vendor needs to be added to the database.

Licensing: Taskmaster Accounts Payable Capture is an add-on application that requires additional licensing.

1.3.5 Cross industry: Surveys

Although many processes adapt easily to the electronic environment, survey and opinion processing has stubbornly remained a paper-based process. Part of the reason is that a survey is not valid until it has been completed. Also many in-depth surveys are too long to fit easily into an online form. If a survey taker drops out at any step along the way, the survey data becomes useless.

Taskmaster helps marketing companies acquire and process data faster and at less cost by reducing processing time. Also the ability of Taskmaster to handle various unstructured forms enables maximum flexibility in survey form design.

By using Optical Mark Recognition (OMR) technology, Taskmaster captures completed check boxes or bubbles, machine print data, barcode data, and handprint commentary or explanations. It interprets the values and uploads them together with the information of the respondent into a survey database for analysis.

For convenience, surveys can also be scanned or verified in a browser with Taskmaster Web. Taskmaster Web delivers the same functionality for on-site, remote data gathering for conferences, trade shows, and mobile surveyors.

The Survey sample application (see Figure 1-8 on page 28) can be used as an example to configure your own application to process documents. The documents can be as diverse as questionnaires, surveys, tests, evaluations, time sheets, applications, lottery forms, and inventory counts.

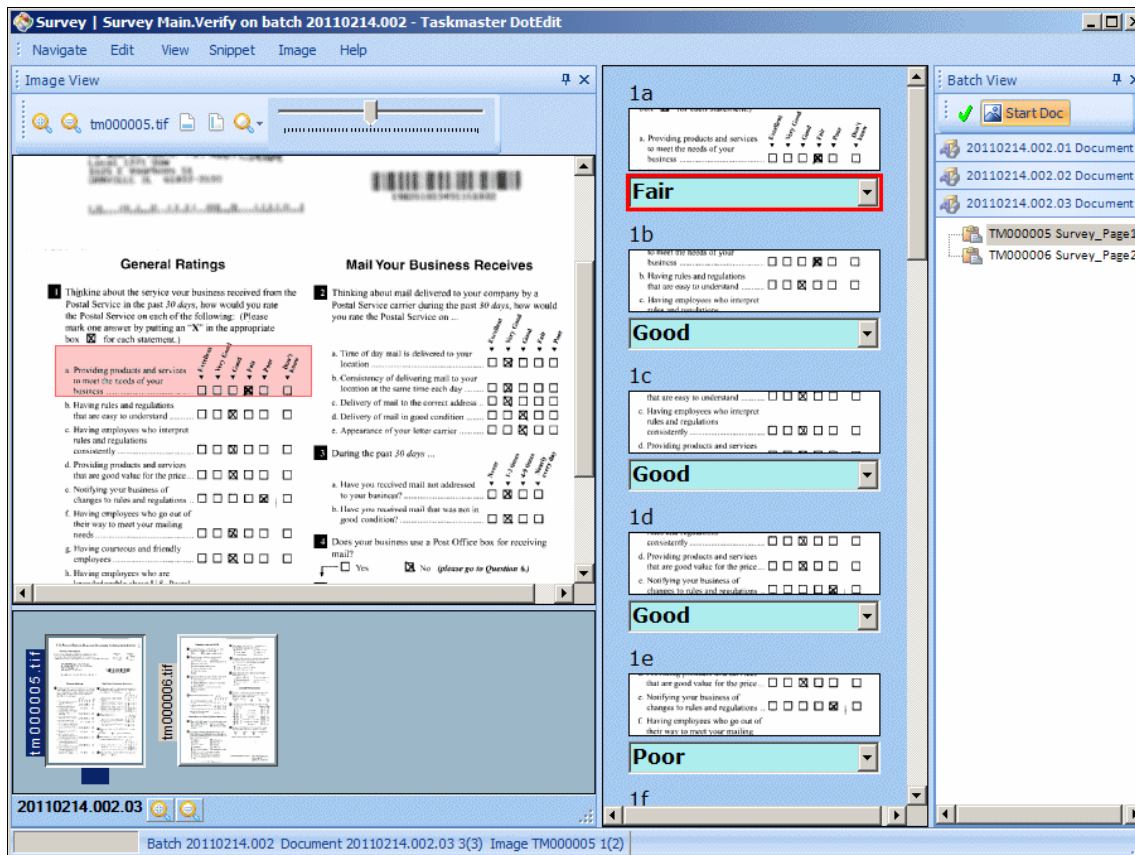


Figure 1-8 Sample application to capture of surveys

1.3.6 Government: Tax return processing

Tax and government revenue organizations rely on document imaging technology to accelerate tax data input, enable easier access to information, streamline operations, and provide better service to taxpayers. Essentially, they can shorten the turnaround time from the receipt of a return to the issuance of a refund check. Although federal, state and local governments encourage businesses and individuals to file returns and other documents electronically, a significant volume of tax forms still arrives at mail centers in paper form.

Taskmaster coordinates the scanning of tax documents and attachments; the extraction, validation, and hand-off of data to back-end systems; and the indexing of images into repositories with minimal human intervention.

Taskmaster Web can enable remote scanning or remote verification by at-home workers to help government tax departments increase productivity and distribute work to lower income areas.

What makes Taskmaster the solution of choice is its ability to handle exceptions and problems that are always associated with high volumes of forms, especially hand-filled forms such as tax documents.

The Taskmaster 1040EZ sample application demonstrates the processing of US tax return forms. The application uses ICR technology to capture hand-written characters (Figure 1-9). It validates the data by checking the presence of mandatory values, applying math, and enforcing verifications against the tax rules, across the entire document.

TaxpayerSSN 012345678		SpouseSSN 		93 257 57	9325757	1TotalWages
012345678				390 00	39000	2TaxableInterest
TaxpayerName Your first name and initial Helen		Last name 			0	3Unemployment
If a joint return, spouse's first name and initial 		Last name 		93 647 57	9364757	4AdjustedGross
Home address (number and street). If you have a P.O. box, see page 12 77113 Main St.		Apt. no. 		blank	X Choice 1	5ParentClaimYE
City, town or post office, state, and ZIP code. If you have a foreign address, see page 12 White Plains, NY 72737				7 200 00	720000	5Exemption
Helen				86 447 57	8644757	6TaxableIncome
77113 Main St.				15 261 27	1526127	7TaxWithheld
White Plains		NY 72737			0	8EIC_C
TaxpayerSignature Helen		SpouseSignature 		15 261 27	1526127	9TotalPayments
Choice 1		blank		19 872 00	1987200	10Tax
					0	11aRefund
				4 610 73	461073	12TaxDue

Figure 1-9 Sample application for capturing a US tax return form

1.3.7 Healthcare and insurance: Medical claims

Insurance companies were among the first to embrace scanning and document management to help control costs and streamline their operations, which depend on the ability to rapidly and accurately process claims and policies. They also need to ensure compliance with Health Insurance Portability and Accountability Act (HIPAA) regulations regarding privacy and data standards.

Taskmaster Medical Claims Capture is a solution that automates data entry for CMS-1500 and UB-04 medical claim forms used in the US. The application helps to eliminate costly, error-prone manual data entry and accelerate claim processing time.

Taskmaster Medical Claims Capture manages the entire capture process, from the scanning of claims to the recognition of data fields to the validation and verification of data for accuracy. It also coordinates the upload of HIPAA-compliant claim data to adjudication systems for payment.

Because Taskmaster Medical Claims Capture (Figure 1-10) is built on the Taskmaster platform, health insurers can take advantage of browser-based distributed scanning and remote indexing for more efficient distribution of work.

The screenshot displays the 'HC_Verify - Batch Pilot' application window. It features a menu bar (File, Edit, View, Navigate, Help) and a toolbar with icons for file operations and navigation. The main interface is divided into several sections for data entry:

- 1a. Insured's ID:** Fields for ID number (954575813) and a secondary ID (954575813).
- 2. Patient Name/Address/Zip Code:** Fields for Last Name (TAYLOR), First Name (TAYLOR), MI, Address (1529 WEST STREET), City (BROOKLYN), State (NY), and Zip Code (11211).
- 3. Patient's DOB:** Fields for DOB (06/11/88) and a secondary DOB (06111988).
- 3. Patient's Sex:** A dropdown menu set to 'Female'.
- 4. Insured Name/Address/Zip Code:** Fields for Last Name (TAYLOR), First Name (TAYLOR), MI, and Address (SAME).
- 11. Policy Number:** Fields for Policy Number (2834318) and a secondary Policy Number (2834318).
- 11a. Insured DOB:** A field for Insured DOB.
- 11a. Insured Sex:** A dropdown menu set to 'Female'.
- 11c. Insurance Plan Name:** Fields for Insurance Plan Name (CLAIMS DEPT) and a secondary Insurance Plan Name (CLAIMS DEPT).
- 11d. Another Plan:** A dropdown menu set to 'Yes'.
- 21. Diagnosis Code:** Fields for Diagnosis Code (466.0 BRONCHITIS ACUTE, 784.0 COUGH) and a secondary Diagnosis Code (786.2 COUGH).
- 24. Service Lines:** A table with columns for 24a. Dates of Service (From-To), 24b. POS, 24c. TOS, 24d. Procedures/Modifiers, 24e. Diag., and 24(f/g) Charges/Days.

24a. Dates of Service (From-To)	24b. POS	24c. TOS	24d. Procedures/Modifiers	24e. Diag.	24(f/g) Charges/Days
01/16/01 - 01/16/01	11		99214	1	170.00 1
01/16/01 - 01/16/01	11		99214	1	17000 1
01/16/01 - 01/16/01	11		71020	1	150.00 1
01/16/01 - 01/16/01	11		71020	1	15000 1
01/27/01 - 01/27/01	11		99213	1	140.00 1
01/27/01 - 01/27/01	11		99213	1	14000 1

Figure 1-10 Medical Claims Capture add-on application showing the capture of healthcare forms

Taskmaster Medical Claims Capture has unique features that claim processor value. For example, it performs automatic database lookups to validate data against provider, member, diagnoses, and procedure codes. It also has an

intuitive verification interface that enables fast and easy identification of claim data.

Because of the higher data density of the medical forms, Taskmaster uses the color dropout technique to remove grid lines in the form background to make it easier to recognize useful data. Taskmaster achieves this processing by using special forms with red ink that drop out when scanned with color filtering or image processing in the scanner.

Figure 1-11 shows the original form on the left side and the color-dropped-out form on the right side. The form on the right side shows only the data that is pertinent to the capture application.

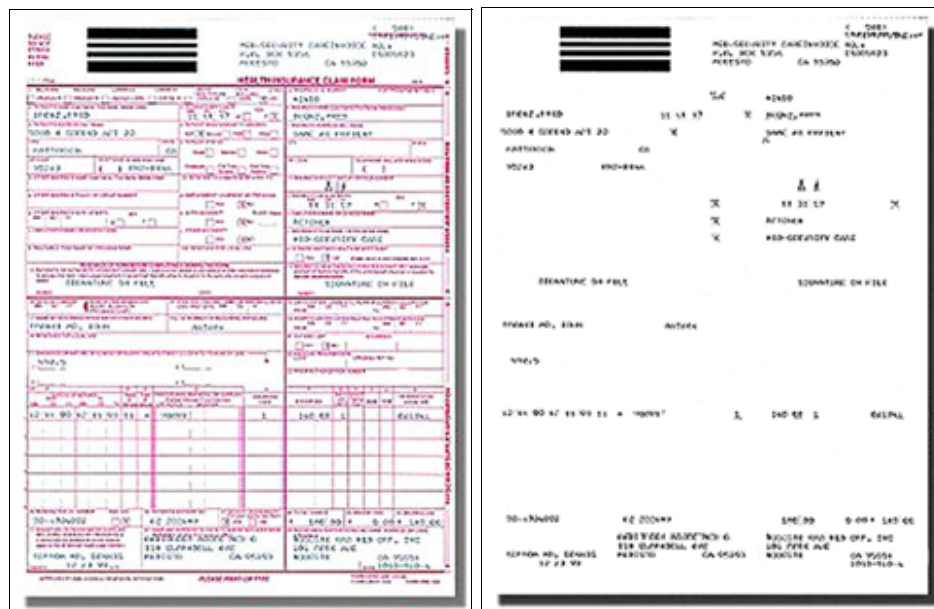


Figure 1-11 Color dropout to remove grid lines

Taskmaster Medical Claims Capture includes the following features:

- ▶ Generates a dynamic template for every page
- ▶ Supports all CMS forms and variations with no pre-configuration of templates
- ▶ Offers the ability for users to capture 100% of fields from both CMS 1500 and UB-04 claims.
- ▶ Captures and stores attachments with an accompanying claim
- ▶ Supports permanent image overlay for archival

- ▶ Offers advanced validations to ensure accuracy, including lookups for member, provider, diagnosis, procedural terminology code, place of service, service date check, and math calculation on charges
- ▶ Enables configuration and modifications without expensive programming
- ▶ Offers HIPAA-compliant 837 EDI export

Licensing: Taskmaster Medical Claims Capture is an add-on application that requires additional licensing.

1.3.8 Banking and finance: Loan applications

Mortgage loan applications can total hundreds of pages of different document types, from titles and credit reports to appraisals to certificates of occupancy. The faster a financial institution can process all the documents required for a mortgage, the faster they can provide funds to the customer. Because mortgage documents must be maintained for the life of the mortgage and beyond, fast and efficient imaging can deliver cost savings and help maintain compliance.

Taskmaster automatically captures data on loan documents to feed back-end systems and accelerate indexing for image storage and retrieval. With the help of content-based classification technology, Taskmaster helps classify and identify all the different documents within a loan portfolio, reducing or eliminating the need for tedious manual prescan preparation.

By using the browser-based Taskmaster Web application, financial institutions can distribute scanning to branch offices and affiliates, so that loan applications can be scanned seconds after final signatures by the customer.

1.3.9 Transportation and logistics: Shipping documents

Transportation and logistics face unique challenges in the gathering and management of data. Transportation companies have a mobile workforce and often a distributed sales force. Tracking deliveries is a paper-intensive process. Also, regulations governing the transport of goods across state lines and between countries are placing increasing constraints on the business.

Companies can use Taskmaster Web for distributed remote scanning of documents generated in the field, such as proofs of delivery, sales orders, and fleet maintenance documents. By using Taskmaster Web this way, companies can reduce the enormous cost of mailing or faxing documents to a central headquarters for processing, accelerate input, and eliminate delays in billing.

Shipping and logistics companies have a growing pressure to provide complete information about the contents of a shipment at the time that the goods cross the border between states or countries. The Patriot Act and other regulations demand rapid and accurate data extraction from complex commercial invoices to fulfill customs requirements. Taskmaster Accounts Payable Capture with its ability to capture line items from multiple page invoices is well adapted to this use (see 1.3.4, “Cross industry: Accounts payable” on page 25).

1.4 Conclusion

As shown in this chapter, Taskmaster can be used in many areas and industries. Bundling Taskmaster with FileNet Content Manager and other tools in the Production Imaging Edition product offering provides *the* all-in-one solution from one vendor for your production imaging needs.

Chapter 2, “System architecture” on page 35, highlights the system architecture of Production Imaging Edition. Then Chapter 3, “Production imaging functionality” on page 51, explains the Taskmaster functionality in more detail, including its user interfaces and the capture process.



System architecture

This chapter describes the architecture and deployment models of IBM Production Imaging Edition with an emphasis on IBM Datacap Taskmaster Capture (Taskmaster).

This chapter includes the following sections:

- ▶ Architecture overview of Production Imaging Edition
- ▶ Components of the Taskmaster system
- ▶ Components of FileNet Content Manager
- ▶ Overall system architecture
- ▶ Deployment of Production Imaging Edition
- ▶ Conclusion

2.1 Architecture overview of Production Imaging Edition

The Production Imaging Edition offering consists of Taskmaster, FileNet Content Manager (including the business process management (BPM) tools), FileNet Workplace XT, and Daeja ViewONE Pro. Figure 2-1 illustrates these components and their interactions at a high level. From top to bottom, you see the client layer, the application logic and presentation layer, and the data layer. The Taskmaster components are highlighted in green, and the components delivered with the FileNet Content Manager platform are highlighted in yellow.

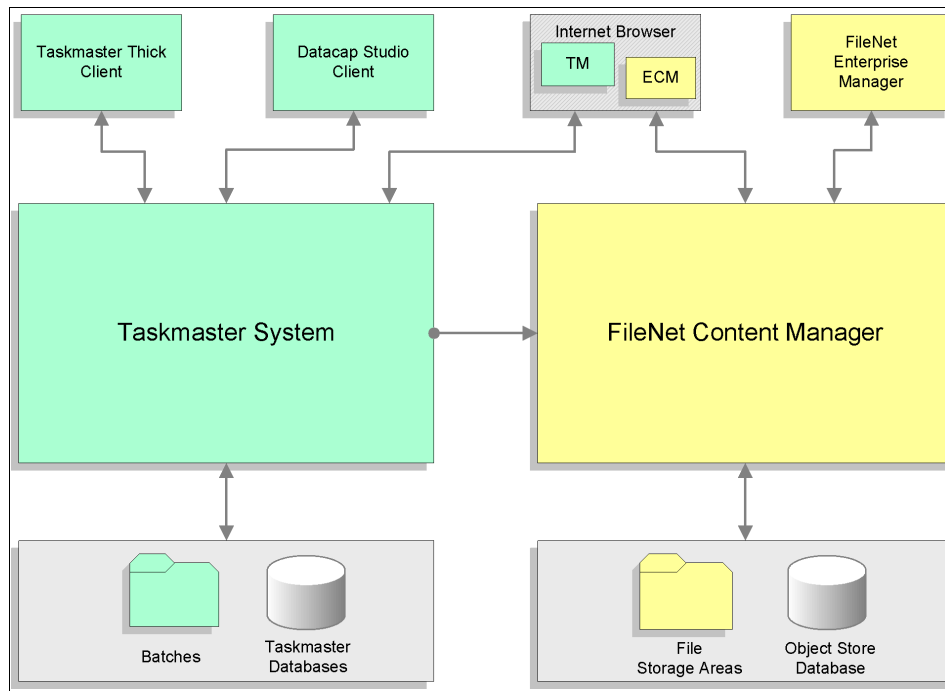


Figure 2-1 Production Imaging Edition architecture

For details about the main components, see 2.2, “Components of the Taskmaster system” on page 37, and 2.3, “Components of FileNet Content Manager” on page 41. For an overall view of the architecture, see 2.4, “Overall system architecture” on page 45.

2.2 Components of the Taskmaster system

The Taskmaster system includes the following components:

- ▶ Business applications or databases
- ▶ Datacap Studio
- ▶ File server
- ▶ Lightweight Directory Access Protocol (LDAP) or Active Directory
- ▶ NENU
- ▶ Rulerunner service
- ▶ RV2
- ▶ Taskmaster databases
- ▶ Taskmaster Server
- ▶ Taskmaster thick client
- ▶ Taskmaster Web Client
- ▶ Taskmaster Web Server

Figure 2-2 shows a detailed view of Taskmaster system architecture.

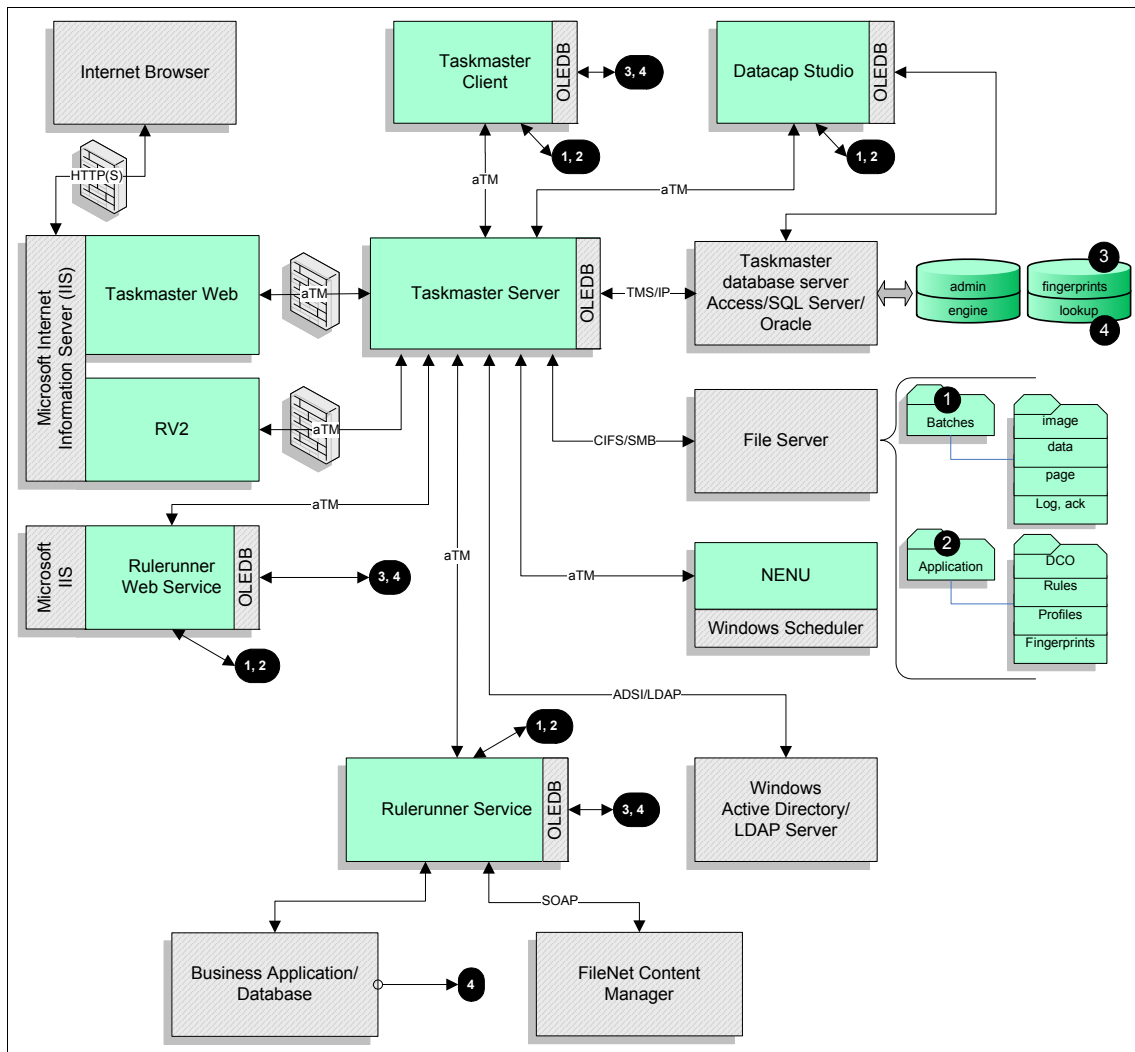


Figure 2-2 Details of the Taskmaster system architecture

This section highlights each component in the Taskmaster system.

Taskmaster Server

The Taskmaster Server is the heart of the Taskmaster system. It manages and serves batches to workstations and users. It also orchestrates the tasks according to the workflow of the Taskmaster application. It provides user authentication and access control, assigns batch IDs, controls batch queuing, and controls access to the Taskmaster databases.

All communications between the Taskmaster Server and its clients, or the other core Taskmaster Server components, use the Datacap Taskmaster socket protocol. For communicating with the databases, it uses Microsoft Object Linking and Embedding for Database (OLE DB). It also uses the Common Internet File System (CIFS) interface to mount the file share that is required to access batches. Taskmaster Server also uses Active Directory Service Interfaces (ADSI) or LDAP to communicate with the Directory Service for user authentication.

Taskmaster databases

For its operation, a Taskmaster application relies on the following relational databases that are hosted in a production system in Microsoft SQL Server or Oracle:

- ▶ The Administrator database stores information about users, groups, workstation, auditing, functional security, and application configuration. It also stores workflow configurations.
- ▶ The Engine database stores information about batches, statistics, and queue states.
- ▶ The Fingerprints database manages the pointers to the fingerprints that are used in a given application.

Each application has its own set of self-contained databases. Also, in many cases, Taskmaster applications need access to other databases to perform lookups or export data to line of business (LOB) systems and databases.

In Taskmaster sample and add-on applications, Microsoft Access database is also used for portability reasons but must not be used in production.

Taskmaster file server

A file server hosts image files, extracted data and control files, and files that are required for running various applications such as the fingerprint files and document hierarchy definition files. The file server must be shared across all the components that need to process the batches.

Taskmaster thick client

The Taskmaster thick client is used to launch user-attended tasks, user and functional security administration, and application configuration. In addition to communicating with the Taskmaster Server, the client requires access to the file server for processing batches (such as viewing a page). It also needs direct access to the fingerprint database and possibly to the lookup databases.

Datacap Studio

Datacap Studio is used to configure the Taskmaster applications, by defining and assembling the document hierarchies, recognition zones and fields, rules, and actions. It requires access to the file server and the Taskmaster databases.

Taskmaster Web

Taskmaster Web interfaces with the Taskmaster Server to serve web pages and documents to web users and to upload documents that are scanned or imported remotely. It is configured as a virtual directory (or site) in Internet Information Services (IIS) for Windows Server. It handles all communications with the back-end services through the Taskmaster Server.

Taskmaster Web client

The Taskmaster Web client runs inside an Internet browser. The ActiveX components are required for scanning and uploading scanned pages to the server and for data entry or verification.

The Rulerunner service

The Rulerunner service runs all tasks that do not require operator intervention, such as image cleaning, conversion, recognition, classification, and export to FileNet Content Manager and databases. It interfaces with FileNet Content Engine through the Web Services application programming interface (API).

RV2

RV2 is a web application that is used to display Taskmaster reports on system activities such as batch status, station activity, or problem batches.

NENU

NENU is used to automate system health and housekeeping tasks. Such tasks include batch monitoring, status notification, and automatic deletion of completed batches. Tasks are scheduled by using the Microsoft Windows Scheduler.

Connection to a business application or database

Typically, connection to a business application or database is through Open Database Connectivity (ODBC). For example, a customer, vendor, or purchase information can be queried against a database and used for image process verification purposes. In addition, information that is extracted from an image can be exported to business applications and databases.

Connection to an LDAP or Active Directory service

An LDAP or Active Directory service is also often part the configuration for Taskmaster users to authenticate.

2.3 Components of FileNet Content Manager

FileNet Content Manager bundled with Production Imaging Edition consists of the following components:

- ▶ Content Engine
- ▶ FileNet Enterprise Manager
- ▶ Process Engine
- ▶ Case Analyzer
- ▶ Process Simulator
- ▶ Business Process Manager Tools
- ▶ Daeja ViewOne Pro viewer

Figure 2-3 shows a detailed view of the FileNet Content Manager architecture.

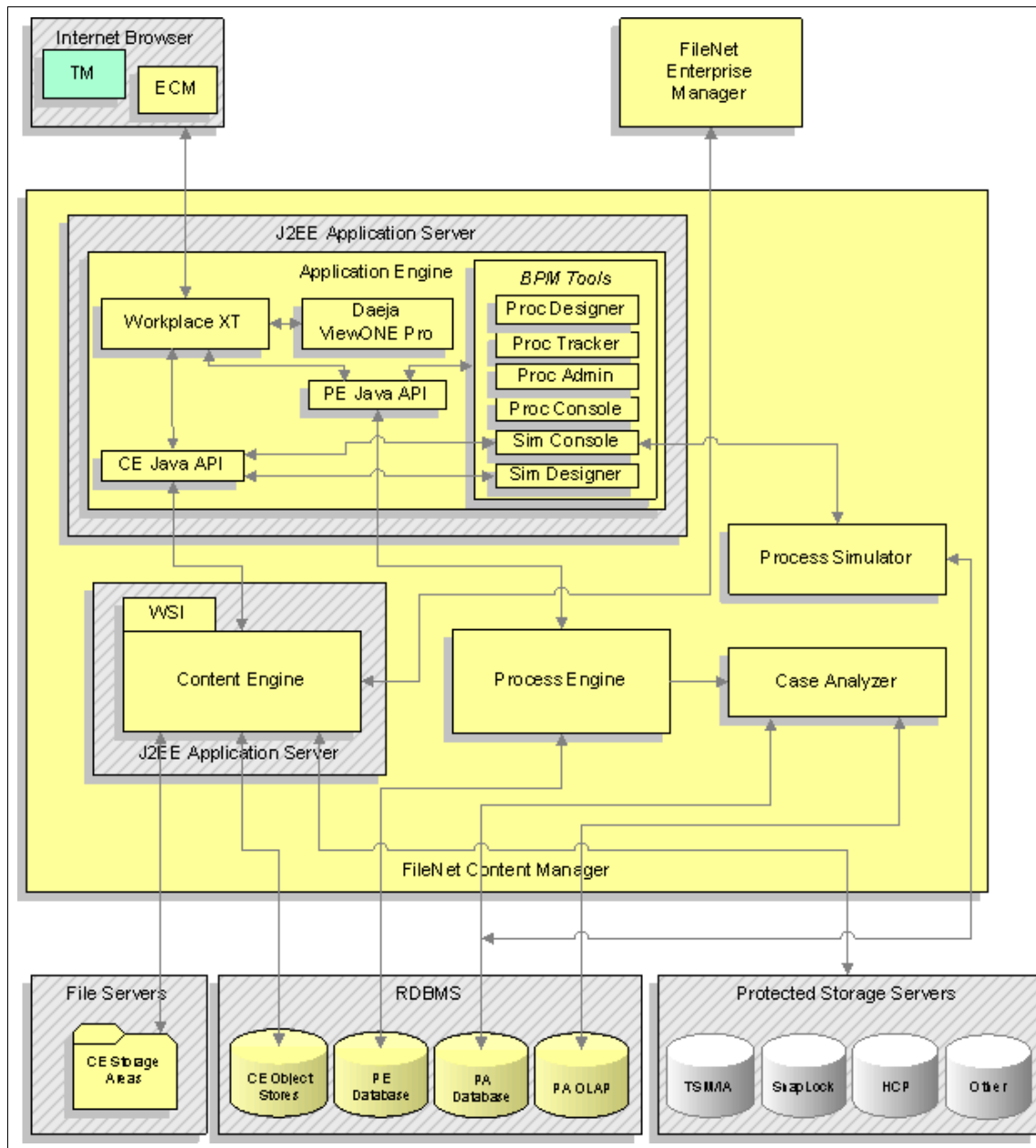


Figure 2-3 FileNet Content Manager architecture

This section highlights each component of FileNet Content Manager.

Content Engine

The Content Engine runs in a Java 2 Platform, Enterprise Edition (J2EE), application server. It interfaces with the object stores in Microsoft SQL Server, Oracle, or IBM DB2® to store the document metadata. It also interfaces with File Storage Areas or Fixed Storage Areas to store document content files. In addition, it interfaces with the LDAP server for the administration of users.

FileNet Enterprise Manager

The FileNet Enterprise Manager is used for the administration and configuration of FileNet Content Manager. It runs as a thick client application that connects to the Content Engine by using its COM API.

Process Engine

The Process Engine runs the workflows that route image documents that are stored in the Content Engine to users. It typically runs on its own machine and connects to a Microsoft SQL Server, Oracle, or DB2 database to store runtime workflow data and queue states.

Case Analyzer

The Case Analyzer interfaces with the Process Engine to accumulate statistical information about workflows in an Online Analytical Processing (OLAP) cube in Microsoft SQL Server.

Process Simulator

The Process Simulator interfaces with the Process Analyzer database to use historical data. Alternatively, it interfaces with the workflow definitions in the Content Engine to run simulations.

Business Process Manager Tools

The BPM Tools consist of the Process Designer, Process Tracker, Process Administrator, and the Simulation Console and Designer. They are all Java-based applications that run on the Application Engine of FileNet Content Manager as a thin client.

Daeja ViewONE Pro

Daeja ViewONE Pro is used by FileNet Content Manager users who run FileNet Workplace XT in Microsoft Internet Explorer to display, annotate, and redact documents. Daeja ViewONE Pro also offers additional servlets in addition to the standard version for performing redaction and streaming. These additional servlets require extra licensing.

Figure 2-4 shows a diagram of the Daeja ViewONE redaction and streaming architecture.

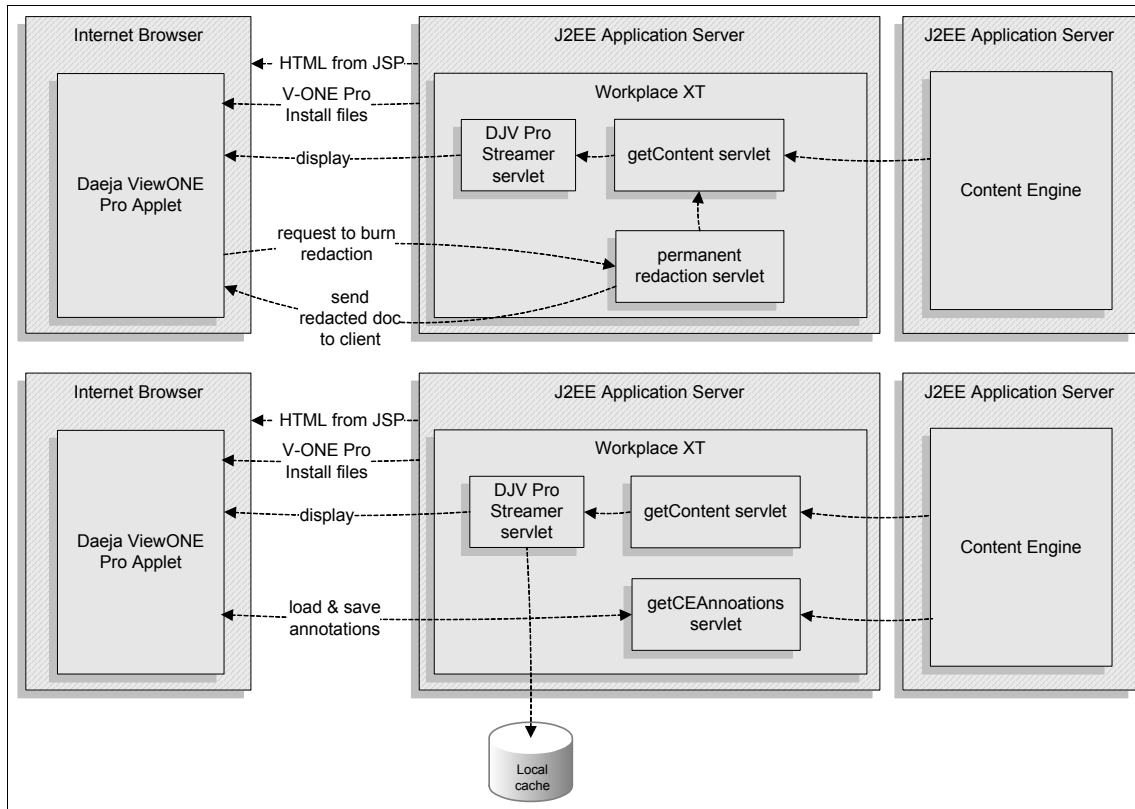


Figure 2-4 Daeja ViewONE redaction and streaming architecture

2.4 Overall system architecture

Figure 2-5 shows the overall system architecture of Production Imaging Edition.

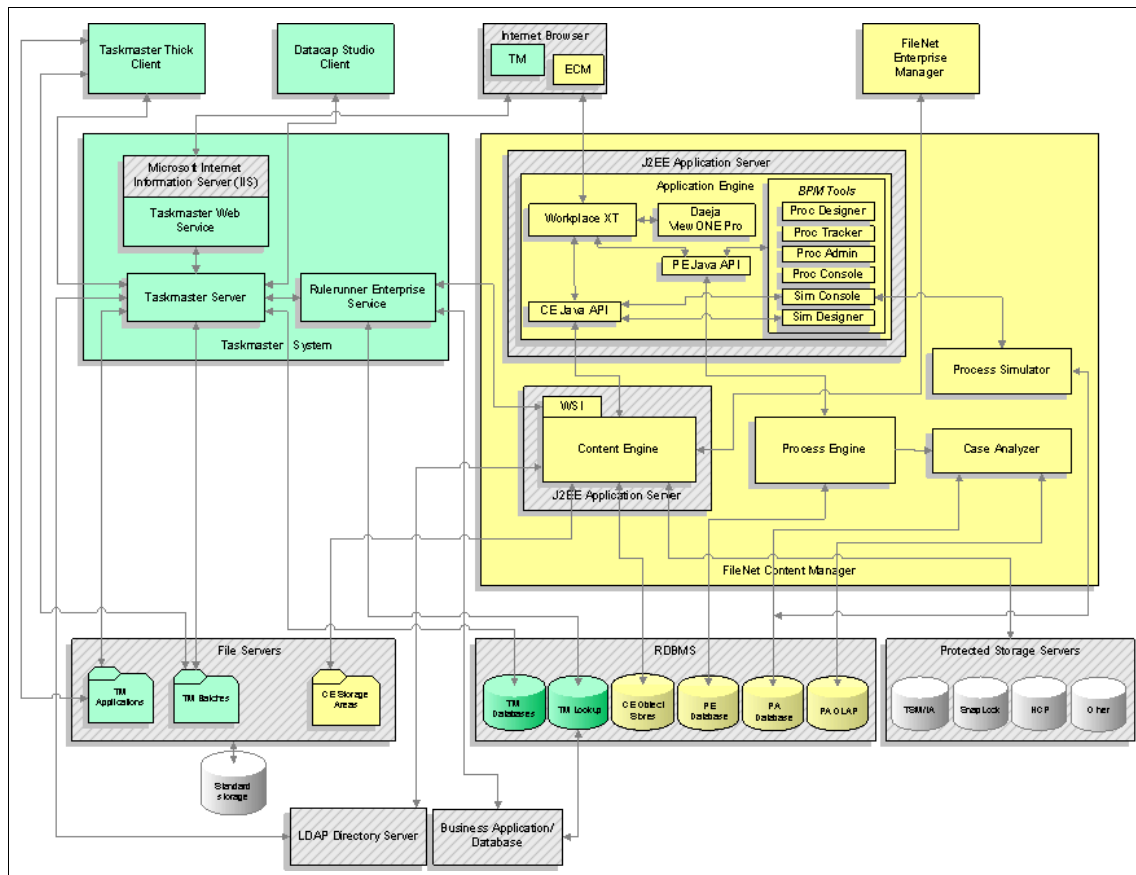


Figure 2-5 Overall architecture of Production Imaging Edition

For more information about FileNet Content Manager, FileNet P8 Platform in general, or FileNet Business Process Manager (BPM), see the following IBM Redbooks publications:

- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547
- ▶ *IBM FileNet P8 Platform and Architecture*, SG24-7667
- ▶ *Introducing IBM FileNet Business Process Manager*, SG24-7509

2.5 Deployment of Production Imaging Edition

Production Imaging Edition combines the deployment capabilities of FileNet Content Manager and Taskmaster to provide flexible deployment models. These models include centralized deployment, distributed deployment, and web deployment.

2.5.1 Centralized deployment

Centralized deployment is where all the production imaging operations, including capture and content management, are done in one location. This type of deployment is used typically when operations need to be concentrated in one place, such as in a traditional mailroom scenario. This approach is best suited when incoming image volumes are high and when economies of scale can be derived from pooling resources and specializing operators to specific tasks, similar to an assembly line.

For example, in a mailroom, high-end production scanners can be used to handle the bulk of the scanning volumes of an organization. Paper documents are collected and shipped to the central location where they are scanned, processed, and exported to FileNet Content Manager. After the documents are ingested in FileNet Content Manager, users get their work items and documents over the wide area network (WAN) or over the local area network (LAN) if they are at the central site.

Figure 2-6 shows an example of a centralized deployment.

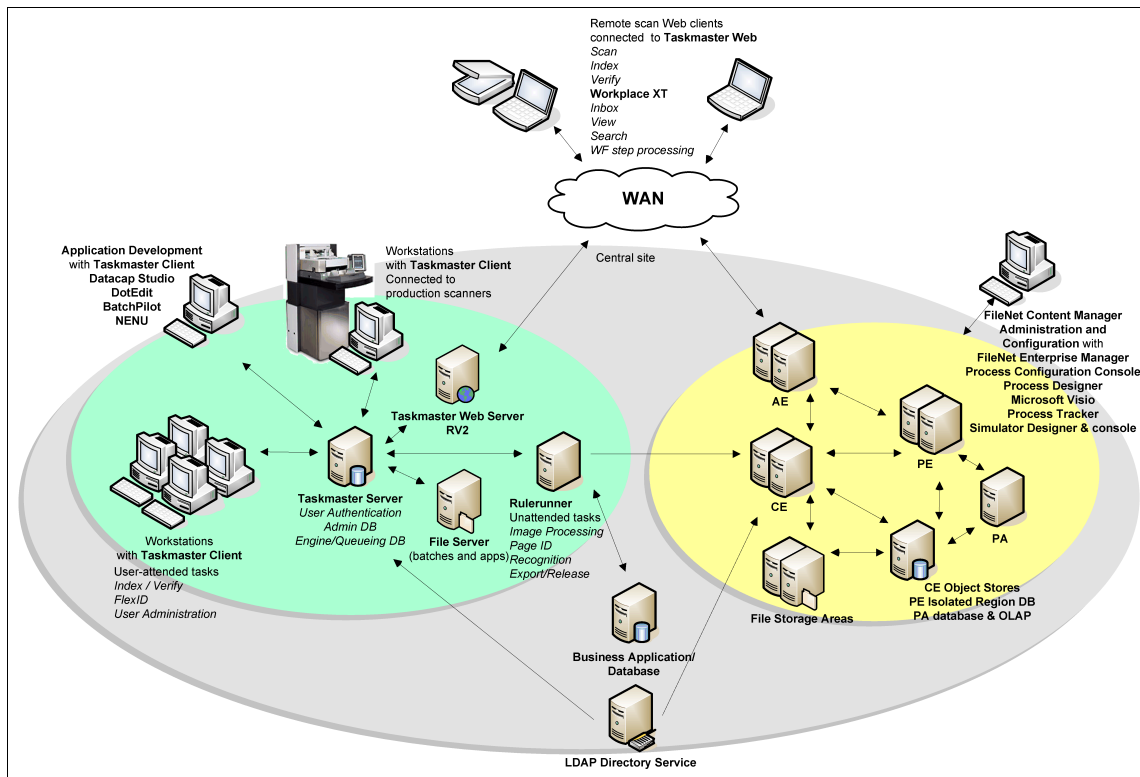


Figure 2-6 Centralized deployment

2.5.2 Distributed deployment

A distributed deployment configuration is ideal for organizations with geographically dispersed user populations and resources and where key system resources can be located closest to users. A distributed deployment can be thought of as a variant of the central deployment model. In a distributed deployment, a large population of users and sizable capture operations justify installing system resources in regional offices.

For example, in a regional office, you might want to install an Application Engine and a Content Engine with content caching, an instance of Taskmaster, and a departmental scanner.

The diagram illustrates a distributed document management system architecture. A central site is connected via a Wide Area Network (WAN) to several remote sites.

Central Site Components:

- Taskmaster Web Server RV2**: Connected to the WAN and the Taskmaster Server.
- Taskmaster Server**: Manages User Authentication, Admin DB, Engine/Queueing DB, and connects to the File Server.
- File Server (batches & apps)**: Connected to the Taskmaster Server and Rulerunner.
- Rulerunner**: Manages Unattended tasks, Image Processing, Page ID, Recognition, and Export/Release.
- Workstations with Taskmaster Client**: Used for User-attended tasks, Index/Verify, FlexID, and User Administration.
- Application Development with Taskmaster Client**: Includes Datacap Studio, DotEdit, BatchPilot, and NENU.
- Workstations with Taskmaster Client connected to production scanners**: Used for Remote scan, Web clients, Scan, Index, and Verify.
- Workplace XT**: Used for Inbox, View, Search, and WF step processing.
- Business Application Database** and **LDAP Directory Service**: Connected to the central site.

Remote Site Components:

- Remote site (Yellow Area)**: Contains **AE** (Application Engine) and **CE** (Content Engine) servers, a **cache**, and a **File Storage Area**. It also includes **CE Object Stores** for PE Isolated Region DB and PA database & OLAP.
- Remote site (Green Area)**: Contains **Taskmaster Client** for user-attended tasks, a **Taskmaster Server**, and a **File Server**.
- Remote site (Yellow Area)**: Contains **AE** and **CE** servers, a **PE** (Process Engine), and a **PA** (Process Administrator).
- Remote site (Green Area)**: Contains **FileNet Content Manager** for Administration and Configuration, **FileNet Enterprise Manager** for Process Configuration Console, Process Designer, Microsoft Visio, Process Tracker, and Simulator Designer & console.

Network and Data Flow:

- The **WAN** connects the central site to the remote sites.
- Request forwarding** is shown between the central site and the remote sites.
- Data flows between the central site and the remote sites, including **Request forwarding** and **File Storage Areas**.

After the documents are ingested into FileNet Content Manager, users in the regional office can connect to the local Application Engine. In turn, the Application Engine transparently communicates with the remote Process Engine (at the central site) to get their work items. With the local Content Engine cache, the users can get the images that have been scanned locally and released by Taskmaster to the local Content Engine instance. The images stay in cache for local consumption while they are written to the file storage areas at the central site. This setup provides the best performance for local users because images do not need to be transferred over the WAN.

48 Implementing Imaging Solutions with Production Imaging Edition and Datacap Taskmaster Capture

For more information about the deployment options of the FileNet P8 platform, see *IBM FileNet P8 Platform and Architecture*, SG24-7667.

2.5.3 Taskmaster Web deployment

Flexibility can be added to the previous two deployment scenarios with Taskmaster Web. Taskmaster Web provides close to the same functionality that is available in its thick client, including scanning, importing, indexing, and verifying documents, in addition to administering the Taskmaster system. Essentially, all user-attended functions of the typical Taskmaster process can be performed through a browser.

For example, by adding Taskmaster Web to the deployment scenarios described earlier, you can perform the following tasks:

- ▶ Supplement the indexing and verification operations for documents that have been scanned at the central location by using resources in remote locations. This task is ideal in situations where remote users are most familiar with the content being processed or where additional assistance is required to handle peak scan volumes.
- ▶ Distribute “document-at-a-time” scanning at the source, while having the indexing and verification done centrally. For example, in this scenario, shipping personnel scan documents locally, while the indexing and verification are executed centrally, where more customer information might be available from the LOB systems.
- ▶ Offload all scanning, indexing, and verifying operations to the local offices. These offices have all the information necessary for these operations and are most likely to use the documents after they are committed to FileNet Content Manager. This task is possible considering that the volumes for each individual are manageable. In this scenario, you do not need many resources in the central location beyond simply monitoring and maintaining the systems.

2.6 Conclusion

This chapter described the system architecture of Production Imaging Edition and its components. It also highlighted the various deployment options.

Chapter 3, “Production imaging functionality” on page 51, focuses on the capabilities, user interfaces, and internal processes of Taskmaster.



Production imaging functionality

This chapter highlights the functionality provided in the IBM Production Imaging Edition offering. It focuses on the capabilities, user interfaces, and internal processes of IBM Datacap Taskmaster Capture (Taskmaster).

This chapter focuses on Taskmaster functionality, from a high-level gradually toward a more detailed level. Concepts and capabilities are introduced as part of a typical Taskmaster process, from digitization, to data capture, to committing contents and data to the back-end systems.

This chapter includes the following sections:

- ▶ Functionality highlights of Taskmaster
- ▶ Taskmaster process
- ▶ Taskmaster GUI
- ▶ Taskmaster clients
- ▶ Taskmaster background processes
- ▶ Taskmaster action libraries
- ▶ Principles and tools of the Taskmaster configuration
- ▶ FileNet Content Manager for production imaging
- ▶ Advanced production imaging viewing
- ▶ Bulk Import Tool
- ▶ Conclusion

3.1 Functionality highlights of Taskmaster

Taskmaster is document capture software. It is bundled into IBM Production Imaging Edition to provide the advanced document capture capability, which is an important requirement in managing the entire document imaging life cycle. The purpose of Taskmaster is to digitize paper documents, extract useful information from them, and feed them into other business processes downstream. Its strength is its ability to perform these tasks with a high degree of automation and accuracy.

Taskmaster provides the following main functions:

- ▶ Scans paper documents, straight from a scanner
- ▶ Imports electronic documents or existing images from a file system or fax server
- ▶ Cleans up images, using such functions as deskewing, removing lines, smears, and borders, to help with the recognition process later
- ▶ Classifies and separates types of document

This way it can determine where data needs to be extracted from the paper and which data needs to be extracted for indexing or any other use in the future.

- ▶ Extracts data by using recognition technologies

The following recognition technologies can be used:

- Optical Character Recognition (OCR) for machine-printed characters
- Intelligent Character Recognition (ICR) for handwriting, typically detached block letters
- Optical Mark Recognition (OMR) for catching check boxes and other expected marks, such as bubbles in surveys or a signature on a form

Taskmaster can also extract data from one- and two-dimensional barcode types. One-dimensional (1D) barcodes contain simple product or item information such as the barcodes around the corner of product boxes purchased in a store. Two-dimensional (2D) barcodes usually contain more information such as address and shipping information.

- ▶ Checks the accuracy of extracted information and corrects errors

Taskmaster can format and validate field values against business rules. It can also automatically look up information in a database from the partial data it has recognized. Taskmaster can trigger verification and validation by a human operator when confidence in the data accuracy is below a set level.

- ▶ Learns automatically from the experience of human operators and the processing of documents to improve accuracy over time
- ▶ Exports image documents and extracted data to FileNet Content Manager and to a database or a business application
- ▶ Organizes the flow of tasks in the capture process from scan to export and exceptions in a workflow
- ▶ Controls access to the system and tasks by using functional security
- ▶ Monitors progress of capture operations and fixes problems in real time
- ▶ Reports on capture operations and provides statistics on how well the system is performing
- ▶ Runs unattended tasks in the background by using the Rulerunner engine
- ▶ Supports flexible deployment scenarios

Such scenarios can include a central mailroom type of operations and distributed imaging over the web, or regional offices by combining with the distributed deployment capabilities of FileNet Content Manager.

3.2 Taskmaster process

As mentioned in Chapter 1, “Production imaging overview” on page 3, Taskmaster is typically involved in the front end of the production imaging process, or the “precommittal process.” It digitizes and extracts the data that is necessary to classify and index the documents according to the model that has been implemented in the FileNet Content Manager repository and workflow.

To understand the Taskmaster capabilities and how they are used, you must first look at the Taskmaster process and some of its main constructs. At a high level, the Taskmaster process begins with the ingestion of paper documents and finishes with the committal (release) of images and their metadata to the back-end repository and line of business (LOB) databases.

Figure 3-1 shows a typical Taskmaster process.

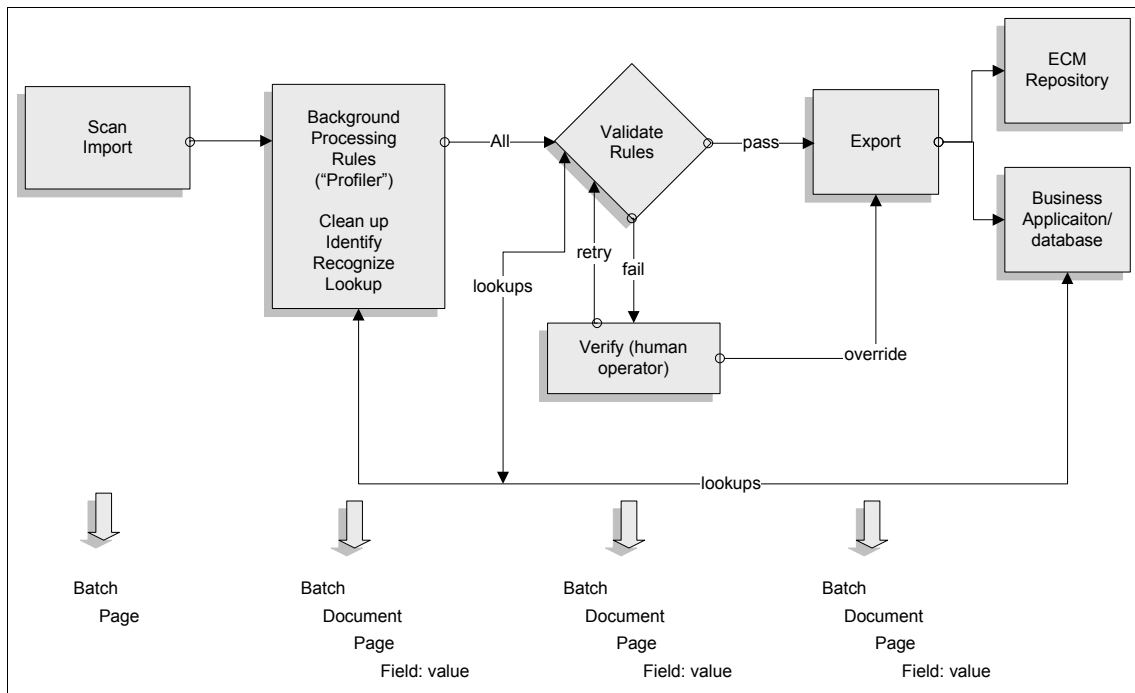


Figure 3-1 Taskmaster process

The main tasks of the Taskmaster process include the Scan task, background processing, the Verify task, and the Export task. The following sections begin with a brief explanation about batch preparation before the Scan task and then provide details about each task.

3.2.1 Batch preparation for scanning

Typically paper documents are prepared manually for scanning. Then they are assembled into batches based on how they are processed later. For example, loan application forms and their supporting documents might be prepared so that they flow in a predefined order that is repeated within a batch.

In another example, when a batch has high variability and little structure, it is useful to insert document separator sheets with a barcode to mark document boundaries. Separator sheets can contain check boxes or other printed data to facilitate the classification process. You can use separator sheets to split a large batch in smaller ones, and you can use a batch cover sheet to automate the indexing in common to all documents in a batch.

3.2.2 The Scan task

When the manual preparation work is complete, the first task in a Taskmaster process is scanning. The Scan task creates a *batch*. The batch is immediately recorded in the Taskmaster Engine database, and a workflow job that controls the processing tasks and sequence is initiated. The scanned images are stored in the Taskmaster file server in individual TIFF files along with an XML description of the contents of the batch.

3.2.3 Background processing task

After populating the batch with images through scanning or importing, Taskmaster processes it automatically in the background. Now *rule processing* comes into play.

Taskmaster processes the batch based on the task definitions or *task profiles*. A task profile specifies sets of rules, or *rule sets*, that Taskmaster must apply in a predefined order to the various components of the batch. These components include the batch itself, documents, pages, and fields or zones. The relationship between these various components is defined in the *document hierarchy* when configuring the Taskmaster *application*.

As Taskmaster processes the batch, it writes the processing results to the batch directory. It also creates a batch descriptor file that shows the name of the task, which is used to pass on information to the next task.

The background processing task can execute many different sets of rules depending on the specifics of the application. Typically it includes the following rule sets:

- ▶ Cleaning and enhancing the images to help the recognition process
- ▶ Identifying the class of document by using *fingerprints*, barcodes, pattern matching, keyword searching, or a combination of these techniques
- ▶ Recognizing the data from the zones and fields defined in the document hierarchy
- ▶ Validating accuracy, formatting, consistency, and completeness of the recognized data and looking up missing pieces of information from external systems

At the end of the background processing, a batch description file is created that describes the batch, its documents, and the pages. The file also links to the XML files that hold the metadata that is extracted from each image.

3.2.4 The Verify task

The background process flags images that have a low recognition *confidence level* or validation errors and that need to be verified by an operator. Taskmaster launches the graphical user interface (GUI) that has been configured for the Verify task. It points the operator to the problem fields with the matching *image snippet*. The operator manually updates the data and submits the changes for validation with the rules that have been associated with the fields. The process is repeated for each problem field until the data is successfully validated or possibly overridden by the operator.

At the end of the Verify task, again, a batch description file is created with the files that contain the verified data.

3.2.5 The Export task

After the images are processed through the background and verification tasks, Taskmaster exports the image files and extracted data to the back-end systems. The Export task includes all the processing that is needed to prepare the documents (possibly converting them to another format), establishes the connection with the back-end systems, and indexes and uploads all the documents in the batch.

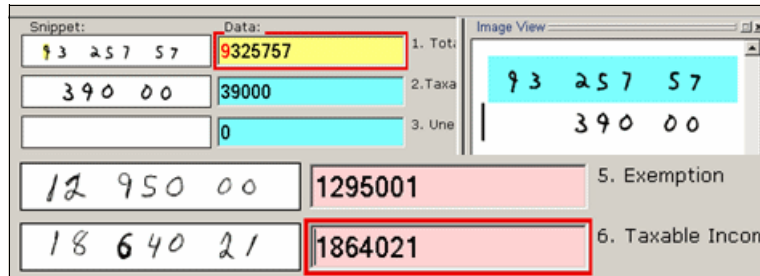
Again, the processing is defined as sets of rules, and they are executed on the batch, document, page, and field levels in the background.

3.3 Taskmaster GUI

Taskmaster provides a flexible user interface, especially for highly customized data entry forms, with innovative capabilities to enhance the productivity and experience of an operator.

3.3.1 Productive GUI

You can create GUIs with buttons, hot keys, and labels. You can also have the cursor position itself in the next problem fields. This way the operator does not have to look for where to go next. Figure 3-2 shows the user interface with the image snippets and color-coded fields.



Snippet:	Data:	Image View
93 257 57	9325757	1. Tot: 93 257 57
390 00	39000	2. Taxa 390 00
	0	3. Une
12 950 00	1295001	5. Exemption
18 640 21	1864021	6. Taxable Income

Figure 3-2 Taskmaster image snippets and color-coded GUI

3.3.2 Image snippets

Taskmaster provides the capability to position image snippets next to the matching recognized data fields. This way, operators can easily compare recognition results to the zones from where the data originated.

3.3.3 Color-coded recognition confidence

You can configure color-coded backgrounds and character ink to indicate the confidence levels on recognized data and quickly show validation errors. For example, as shown in Figure 3-2, you can use the following backgrounds:

- ▶ Blue background for high-confidence recognition results and no data validation errors
- ▶ Yellow background for low-confidence recognition results with red ink for low-confidence characters in the field
- ▶ Red background for data validation errors

3.3.4 Click'n'Key capability

With the *Click'n'Key* capability, you can pull data into a field with a single click. You position the cursor in the destination field and then click or draw a box around the piece of information in the image that you want to grab (Figure 3-3).

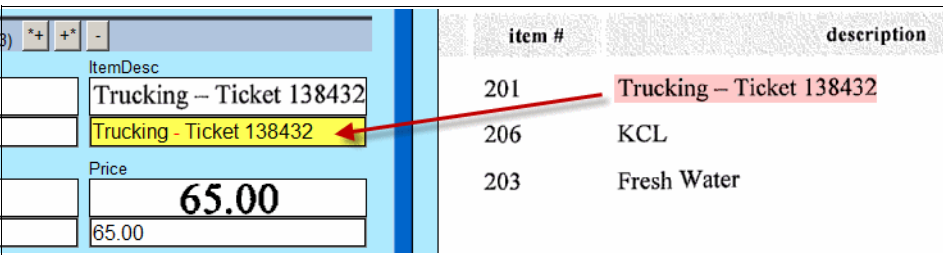


Figure 3-3 Click'n'Key capability in Taskmaster

You can also combine this feature with the ability in Taskmaster to learn from the manual input of an operator and record the location where the data was found in the fingerprint that is associated with the type of document. This new information learned by Taskmaster is used to fine-tune and automate the recognition process the next time it processes a similar document. For more information about fingerprints, see 3.6.7, “Fingerprinting” on page 87.

3.4 Taskmaster clients

Taskmaster can be used with a thick client, for centralized or dedicated capture work, or with a thin client, for remote or more occasional use. This section provides an overview of the functionality of these clients and of their usage to provide a better understanding of the dynamics between the Taskmaster process and its clients.

3.4.1 Taskmaster Client (thick client)

Taskmaster Client is a thick client. It is used by operators for launching tasks (such as the Scan and Verify tasks) and by supervisors for monitoring operations. It is also used for administering users, for workstations, for security, and to configure the *jobs* in the workflow of an application.

From the Taskmaster Job Monitor window (Figure 3-4), a supervisor can monitor the status of each batch in the system and obtain information. For example, a supervisor can gather details about the currently executing task, the elapsed time for the task, who is running the task, on which workstation it is running, and the number of documents and pages in it. The supervisor can also take control of specific problem batches, look up their history, change their status and priority, and assign them a different task, workstation, or user to resolve the issue.

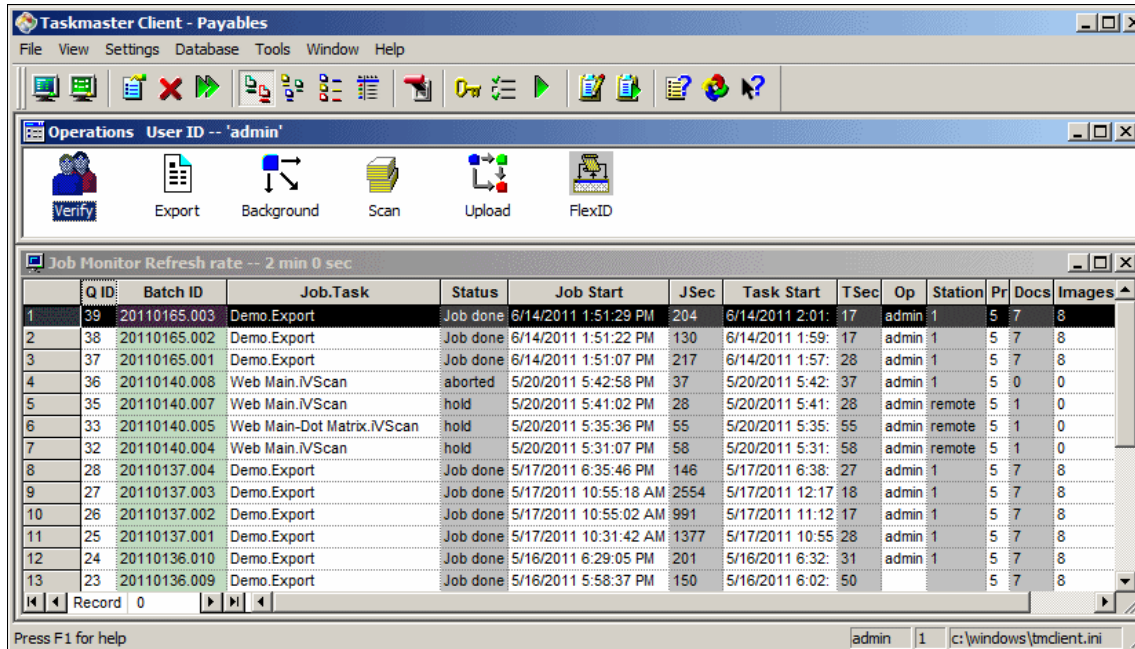


Figure 3-4 Operations and Job Monitor windows in the Taskmaster thick client

When configuring an application, the administrator uses the Administrator window to assemble the set of tasks that were predefined in Datacap Studio (in the task profiles of the workflow of the application). Tasks can run in the background, such as the tasks that are run by Rulerunner. Alternatively, tasks can be user-attended, such as the Verify task. Each user-attended task has a user interface component associated with it, which is specifically for the type of interaction and document in the batch.

A specific arrangement and sequence of elemental tasks constitute a job. Typically each type of scenario has one job. Figure 3-5 shows the Workflow window for Demo with its task hierarchy.

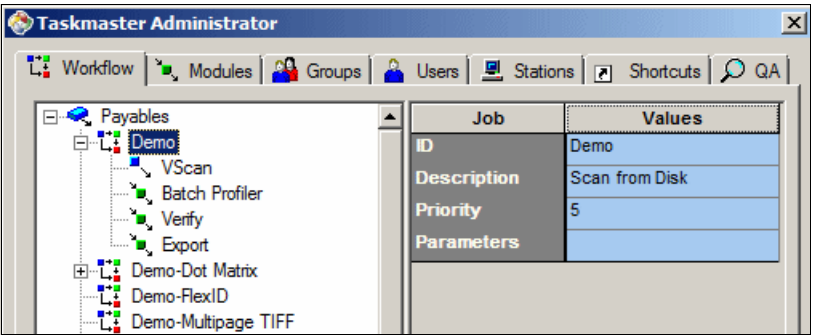


Figure 3-5 Setting up jobs on the Workflow page of Taskmaster Administrator

After the jobs and tasks are defined, they are assigned to users and workstations to provide a tight control over capture operations. The administrator creates workstations, users, and groups and then assigns them privileges (such as monitoring of jobs or workstations) and permissions to run specific tasks as defined in the workflow of the application. Figure 3-6 shows that user Verif1 is given permission to run the VScan and Verify tasks.

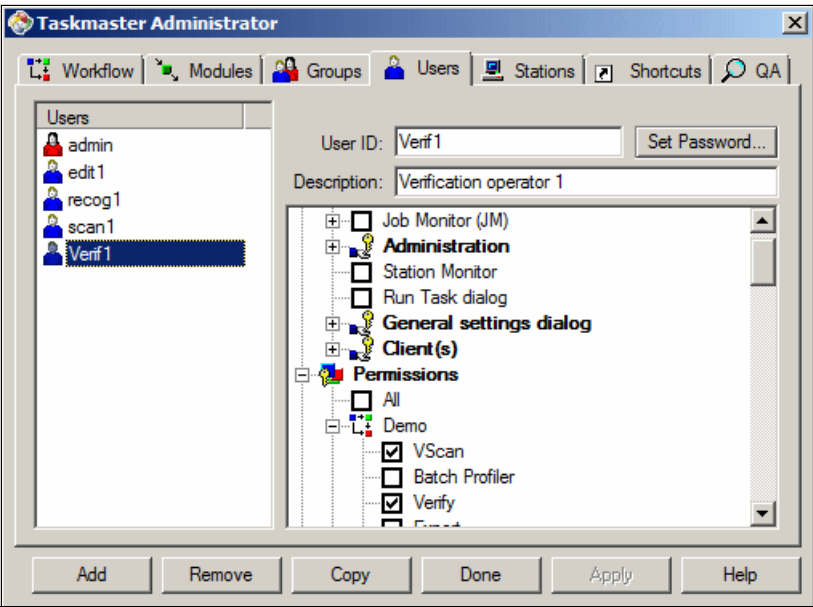


Figure 3-6 Assigning users to tasks on the Users page of Taskmaster Administrator

When user Verif1 launches the Taskmaster Client for the Payable application, only two tasks are available, Verify and Scan, as shown in Figure 3-7.

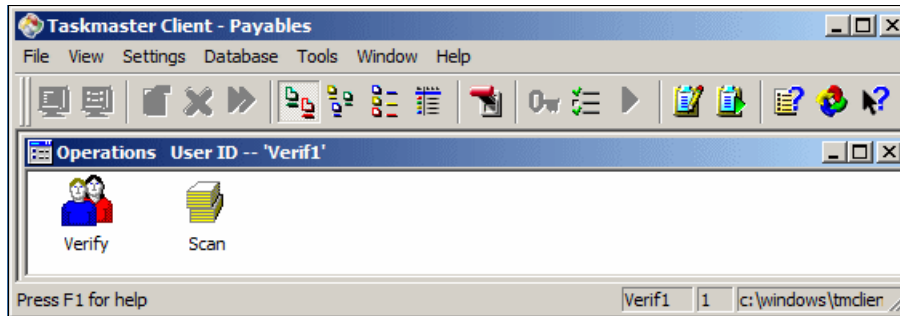


Figure 3-7 Scan and Verify operations assigned to an operator

A task processes an entire batch at a time. The task can be set up to run automatically on the next batch pending in the queue or to allow the user to select it. When a task is set to run automatically, the operator is served with the next batch. Then the process repeats until all outstanding batches for that task have been processed. For more information about tasks, task profiles, and rule processing, see "Rule processing" on page 71.

Batch Pilot for user-attended tasks

User-attended tasks vary depending on the specifics of the application. They are used for scanning, fixing batch issues, and verification.

To design and run the user interface for these tasks, Taskmaster provides a visual tool, called *Batch Pilot*, as shown in Figure 3-8 on page 62. This tool is used at design time to create forms with buttons, snippets, and fields, and the logic behind them. It is called at run time by the Taskmaster Client, according to the configuration done in the Task module of the administrator of the Taskmaster Client. For Verify tasks, forms reflect the particular data structure of the document hierarchy of the application.

In addition to Batch Pilot, Taskmaster includes a new tool, called *Dot Edit*, that dynamically generates runtime windows with data fields laid out in a list. Custom Dot Edit forms can also be designed by using Microsoft Visual Studio.

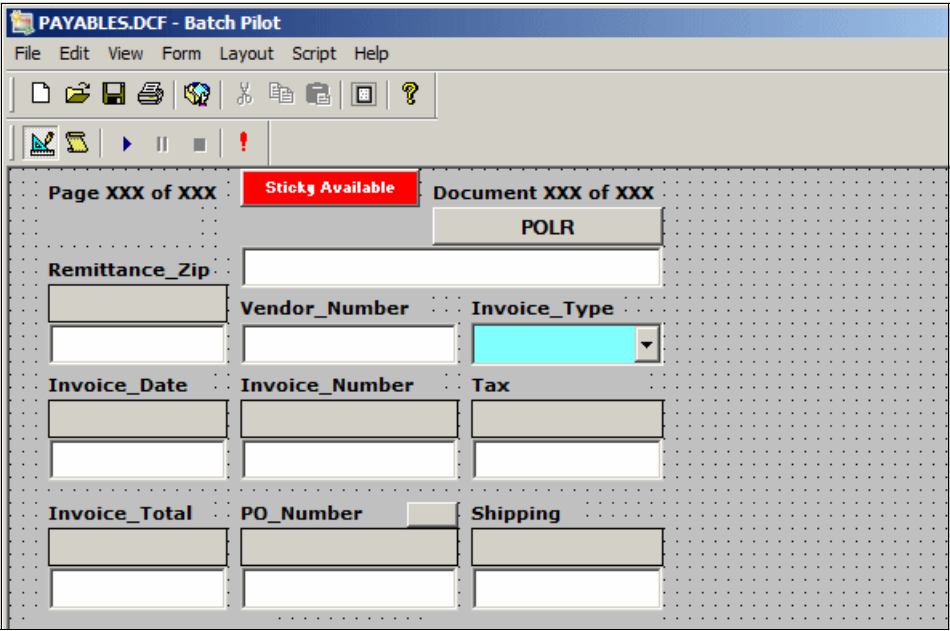


Figure 3-8 Batch Pilot in design mode

Thick Scan task

Taskmaster offers both thick and thin scanning capabilities. The thick client is adapted for high volume scanning, using the Image and Scanner Interface Specification (ISIS) interface. The thick client takes advantage of the rated speed and functions of high-end production scanners, such as duplex and multistreaming scanning. Figure 3-9 shows the Setup window for a thick Scan task.

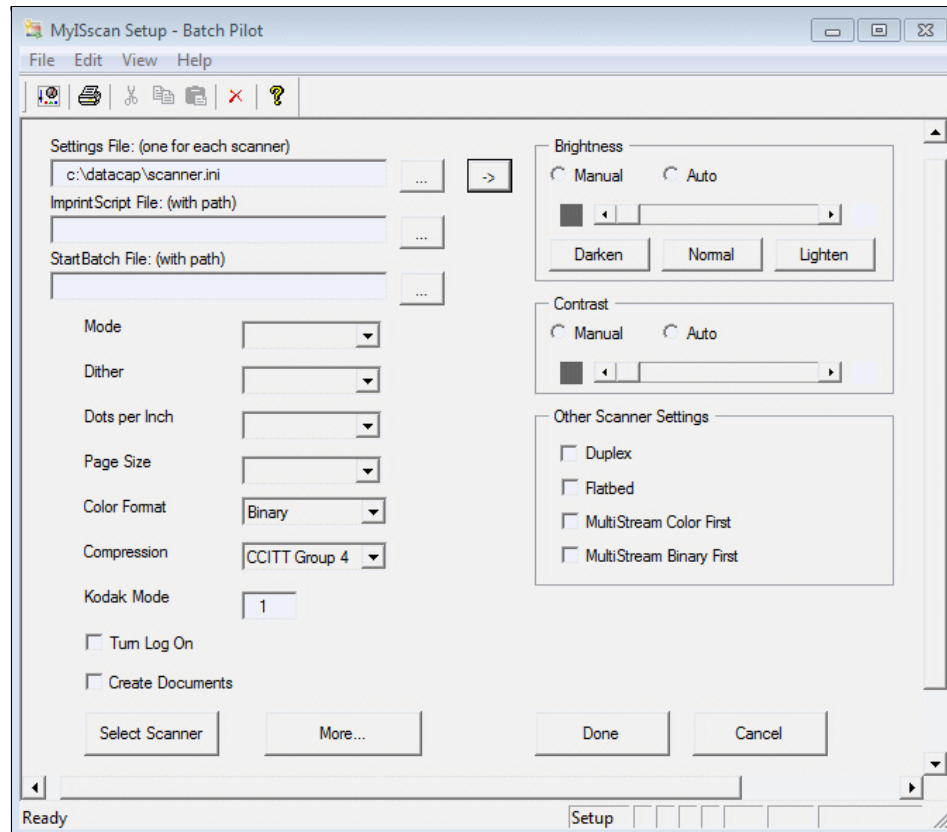


Figure 3-9 Setup window for a thick Scan task

Taskmaster comes with a sample user interface for the Real Scan (rScan) task (Figure 3-10). This window can be customized to the specific needs of the scan operator, by using Batch Pilot, and inserted as the first step in the workflow of the application.

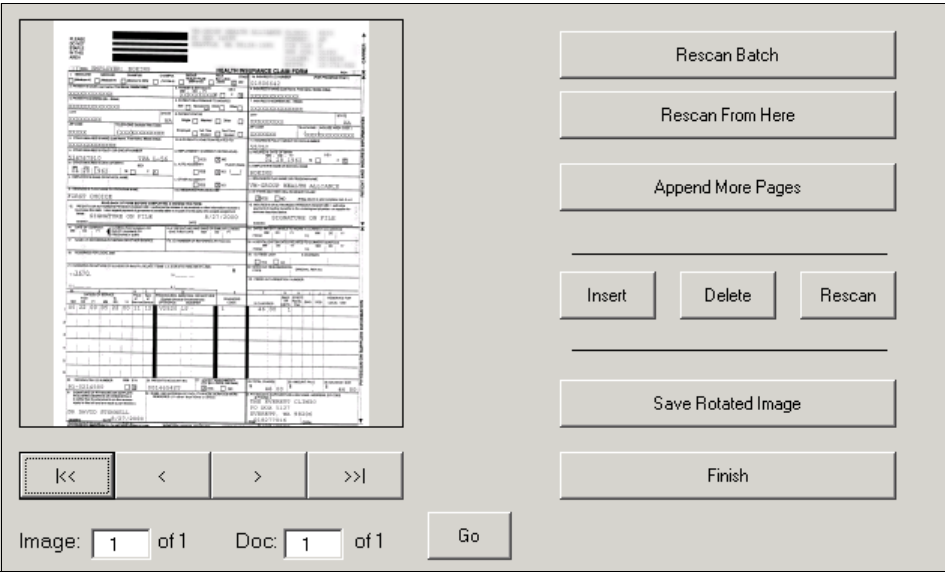


Figure 3-10 Review windows for a thick Scan task

Thick Verify task

Similarly, user interfaces for the Verify task are provided for each sample application that is delivered in Taskmaster. As shown in Figure 3-11, Batch Pilot runs the Verify form shown earlier in the design mode.

Verify - Batch Pilot

File Edit View Navigate Help

Page 1 of 1 Document 1 of 6

Lookup Vendor **POLR (3)**

Remittance_Zip **Comp**

01101 Vendor_Number 999999 Invoice_Type PO

01101

Invoice_Date 02/15/10 Invoice_Number 36149 Tax 107.63

02/15/2010 36149 107.63

Invoice_Total 2157.63 PO_Number 010101 Shipping

2157.63 010101

TTO NEW

Details 1(1)

Linitem 1(5)

Prev Next < Add Add > Del

PO LineNum ItemID Qty

1000-NM 1

1000-NM 1

ItemDesc

000-NM Base Charge

Base Charge

Price LineTotal

2100.00 2100.00

2100.00 2100.00

Calculate Blank Find Details View Details

Image View

Invoice No. 36149

INVOICE

Customer:

Name Techology Co. City Rochester State NY ZIP 14621

Address 100 Business Parkway

Phone (508) 550-1212

Invoice Date 02/15/10

Order No. 700000

PO # 010101

Terms 2/10net30

Qty Item Description Unit Price TOTAL

1	1000-NM	Base Charge	2100.00	2100.00
100	1000-NM	Mileage	9.00	140.00
1	2000-NM	3" Remote Valve	500.00	500.00
1	1000-Add	Additional Man	400.00	400.00
1	Discount	Discount	-100.00	-100.00

Subtotal 2600.00

Ongoing 167.53

Tax 6.25%

TOTAL 2157.53

Office Use Only

Task 20110165.005 20110165.005.01 TM000001

Figure 3-11 Batch Pilot running the Verify task

Figure 3-12 shows the user interface for the same Verify task running in Dot Edit, using a custom form. The customization only applies to the central part of the window where the fields and image snippets are paired up. The other panes of the window, such as the viewer and batch view, are generated automatically.

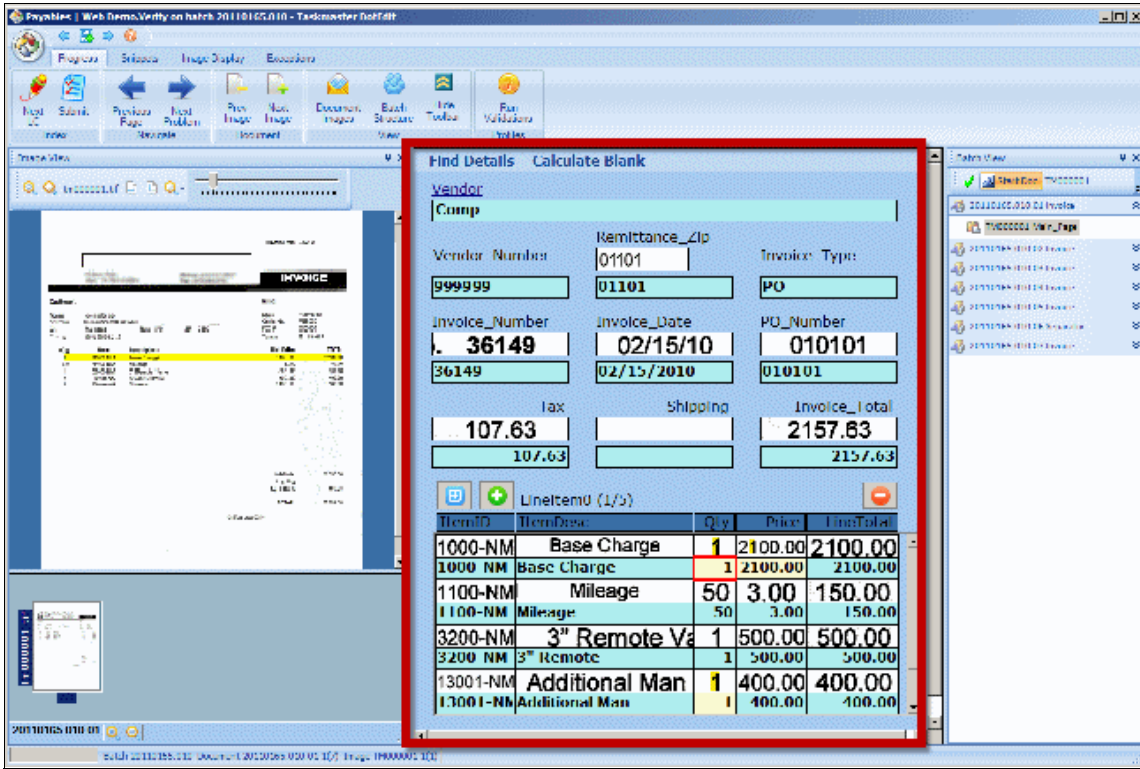


Figure 3-12 Dot Edit running the Verify task

As indicated previously, Dot Edit can also be used in list mode (Figure 3-13). Dot Edit uses the information of document hierarchy, image zones, and data fields already defined for the application in *Datacap Studio*.

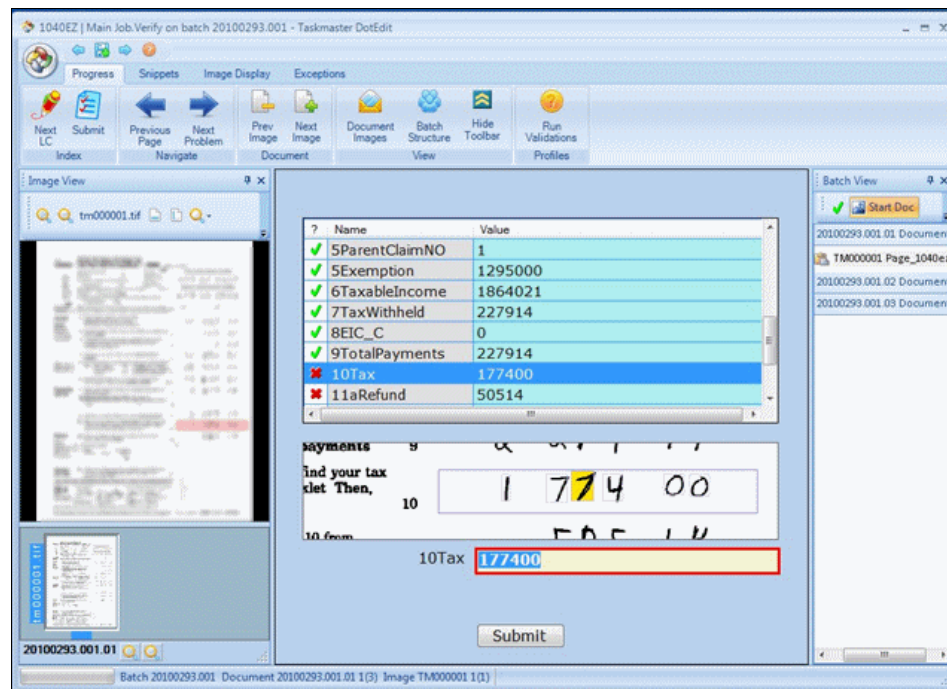


Figure 3-13 Dot Edit automatically generating the user interface of the Verify task

In this case, contrary to a custom form, all fields are laid out as a list. You can edit them one at a time by selecting the field in the list, which opens the field in edit mode and calls the image snippet associated with it. For more information about Datacap Studio, see 3.7, “Principles and tools of the Taskmaster configuration” on page 94.

3.4.2 Taskmaster Web client

Taskmaster Web client offers the same overall functionality as the thick client. It runs tasks, monitors batches, and administers jobs, users, and workstations. Figure 3-14 shows the Job Monitor page of the Taskmaster Web client where you can monitor Taskmaster jobs.

IBM Datacap Taskmaster Capture											
Home Operations Monitor Administrator											
Job Monitor • Station Monitor • Web Monitor											
Items per page 15 Delete batches Filter... Refresh rate --											
QID	Batch	Job	Task	Status	J start	J time	T start	T time	Oper	Station	Pr Docs
41	20110165.005	Demo	VScan	pending	2011-06-14T14:28:53	92	2011-06-14T14:28:53	5			5 7
40	20110165.004	Demo	Export	Job done	2011-06-14T14:28:43	661	2011-06-14T17:10:13	26	admin	1	5 7
39	20110165.003	Demo	Export	Job done	2011-06-14T13:51:29	204	2011-06-14T14:01:49	17	admin	1	5 7
38	20110165.002	Demo	Export	Job done	2011-06-14T13:51:22	130	2011-06-14T13:59:12	17	admin	1	5 7
37	20110165.001	Demo	Export	Job done	2011-06-14T13:51:07	217	2011-06-14T13:57:28	28	admin	1	5 7
28	20110137.004	Demo	Export	Job done	2011-05-17T18:35:46	146	2011-05-17T18:38:16	27	admin	1	5 7
27	20110137.003	Demo	Export	Job done	2011-05-17T10:55:18	2554	2011-05-17T12:17:29	18	admin	1	5 7
26	20110137.002	Demo	Export	Job done	2011-05-17T10:55:02	991	2011-05-17T11:12:37	17	admin	1	5 7

Figure 3-14 Job Monitor page of Taskmaster Web client

Thin Scan task

One difference between the thick and thin client implementations is the support of scanners. A thick client can drive high-end production scanners through its ISIS interface. A thin client requires an ActiveX component to interface with TWAIN drivers. Another difference is that, for thin scan, you might need to insert an Upload task in the job after scanning. This task transfers the scanned document files from the scan directory to the Taskmaster Server over the web if the web server cannot access the scan directory directly. Figure 3-15 shows the Administrator page of the Taskmaster Web client.

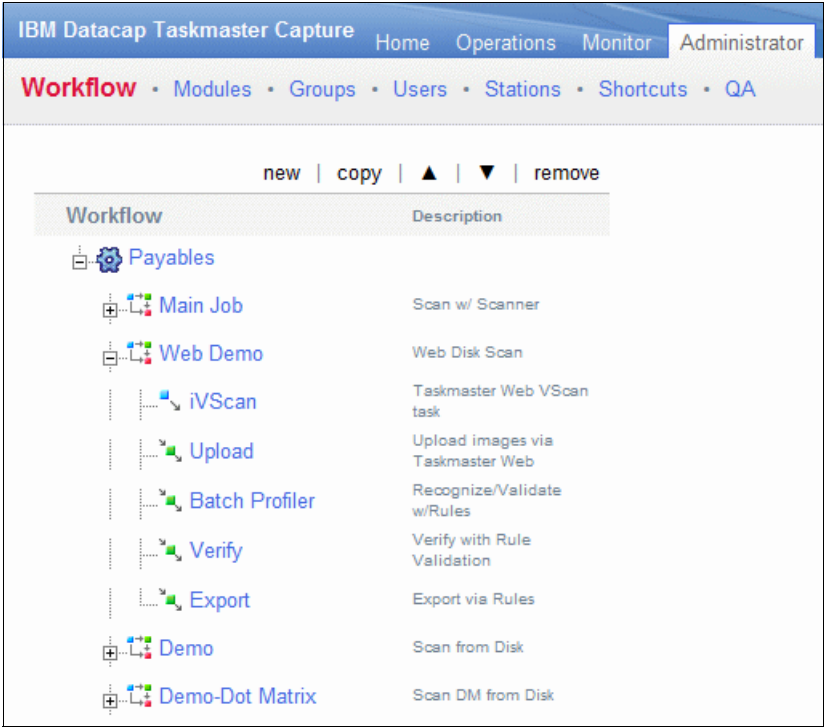


Figure 3-15 Administrator page of Taskmaster Web client

An alternative to driving the scanner directly is to have the scanner write images to an import directory. Then use the Virtual Scan (VScan) task of the thin client to import and manipulate the images in the batch (Figure 3-16) and have the Upload task automatically transfer scanned images to the Taskmaster server.

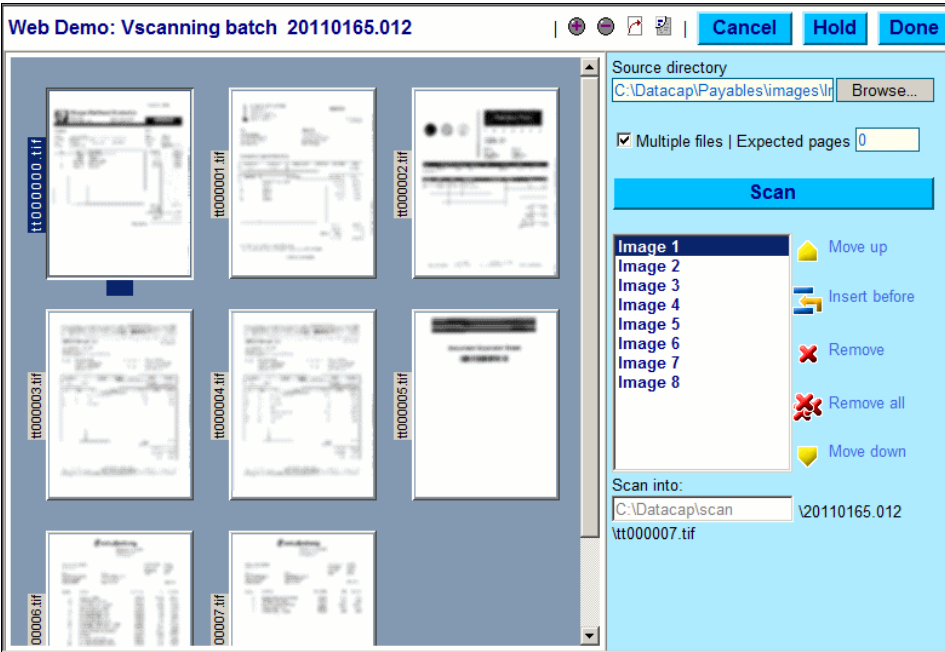


Figure 3-16 Importing from a local directory using Taskmaster Web

The Taskmaster Web user interface offers generally the same productivity features as the thick client does. It includes image snippets, Click'n'Key, and color-coded backgrounds and inks to show confidence levels. The differences are in the implementation, such as adding buttons for validation instead of key short cuts, to account for the nature of the web browser interface.

Figure 3-17 shows the Verify task user interface in Taskmaster Web.

Web Demo.Verify : 20110165.006 Main_Page TM000001 | Batch view...

next LC Disp snip Override Hold Submit

Vendor: SYSTEMS Vendor Number: 444444

Remittance Zip: nd, TX 99999 Invoice Number: DICE #03-8508

99999 #03-8508

Invoice Date: DATE: 02/15/10 PO Number: 1111

02/15/2010 1111

Tax: 27.68 Shipping:

27.68

Invoice Total: 554.68 Invoice Type: PO

554.68

Add New Fingerprint Routing Instructions:

None

Find Details Calculate Blank Submit

INVOICE

SYSTEMS
PO Box 1111
Midland, TX 99999
Phone (333) 333-1000
Fax (333) 333-1001

INVOICE #03-8508
DATE: 02/15/10

To: technologies
9999 Business Parkway
Rochester, MN 22222
(555) 555-1212

Ship To: technologies
9999 Business Parkway
Rochester, MN 22222
(555) 555-1212

Comments or special instructions:

SALESPERSON	P.O. NUMBER	REQUESTIONER	SHIPPED VIA	F.O.B. POINT	TERMS
198	1111				2/10Net30

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Rebuild 2" Motor Valve	175.00	175.00
2	Service	50.00	100.00
80	Mileage	.95	76.00
2	Service	50.00	100.00
80	Mileage	.95	76.00

Figure 3-17 Verify task in Taskmaster Web

3.5 Taskmaster background processes

One of the strengths of Taskmaster is its ability to perform operations on batches in the background. In running the background processes, it relies on the Rulerunner engine and an extensive library of rules and actions.

This section explains how the Taskmaster workflow and rules are processed with the document hierarchy. It also provides an overview of the functionality offered by the Taskmaster actions libraries.

3.5.1 Rule processing

Rulerunner is the process that performs operations, called *actions*, on the objects of a document hierarchy in the background. It can be invoked manually, such as when a human operator validates a field. However, more typically, it is invoked automatically by Rulerunner. Rulerunner is set up to monitor the Taskmaster job queue and execute tasks automatically on batches as they move forward through the Taskmaster process.

Figure 3-18 illustrates how Rulerunner executes rules as specified in each task.

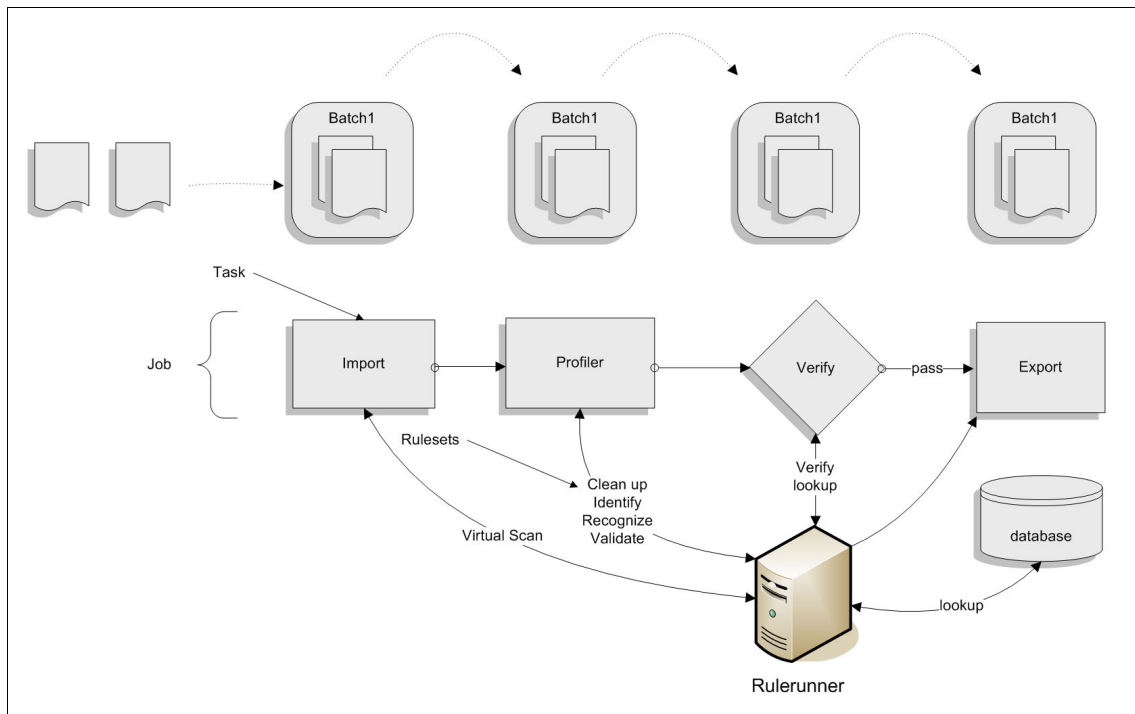


Figure 3-18 Rulerunner executing rules as specified in each task

Taskmaster has two hierarchies. It has a document hierarchy that describes the relationship between a batch, document, page, and field. In addition, it has a workflow hierarchy that describes the relationship between a job, task, rule set, rule, function, and action. Creating a Taskmaster application entails defining these two hierarchies and the interplay between them, as illustrated in Datacap Studio in Figure 3-19.

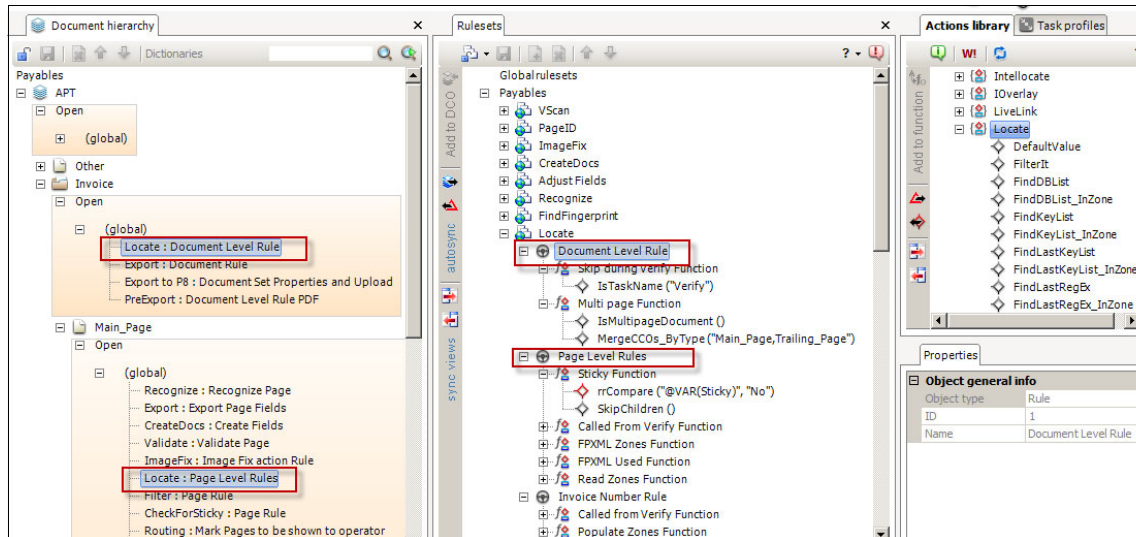


Figure 3-19 Datacap Studio user interface

3.5.2 Job, task, and task profile

A *job* is a particular combination and sequence of tasks in the workflow of an application to address a specific scenario, such as simplex or duplex scanning, thick or web scanning, and so on. When a job is run, Taskmaster creates a batch that is processed from task to task.

When a background processing task is run, Taskmaster invokes the rule sets that were defined in the corresponding task profile. This profile is a template that is used by Taskmaster as an entry point to instantiate a task at run time.

3.5.3 Rule set

A task profile is made up of several rule sets that are arranged in a particular sequence to produce the desired processing results. For example, a task profile can include Cleanup, Identify, Recognize, and Validate rule sets that are run in that order.

3.5.4 Rule

A rule set groups various rules that are bound together to the objects of the document hierarchy of the application and that are executed on demand as Rulerunner walks through the document hierarchy at run time. For example, in Figure 3-19 on page 73, consider the Locate actions that are used to locate words or regular expressions in the recognized data. They can be included in rules and rule sets at the document, page, and field levels, for the specific processing needs of that object. However, the relevant rule of the Locate rule set is only executed upon opening or closing the object of the hierarchy that is associated with it.

More generally, the rules of a rule set can only run when they are mapped to specific objects of the document hierarchy. They can also run only when the rule set they belong to is included in the task profile being executed. The execution order of rules in a rule set is dictated, first, by the order in which the parent rule set appears in the task profile and, second, by the processing sequence of the objects in the runtime document hierarchy.

3.5.5 Processing of the document hierarchy at run time

When a task is invoked, Rulerunner processes recursively each object that is found in the runtime batch (Figure 3-20 on page 75). It starts at the batch level and proceeds to separate the first document, then the first page in it, then all the fields in that first page, then the next page, and so on. It repeats this process with the next document. As it processes each object in this manner, it calls the rule sets that are bound to it. Rule sets can be configured to execute on opening or on closing the object.

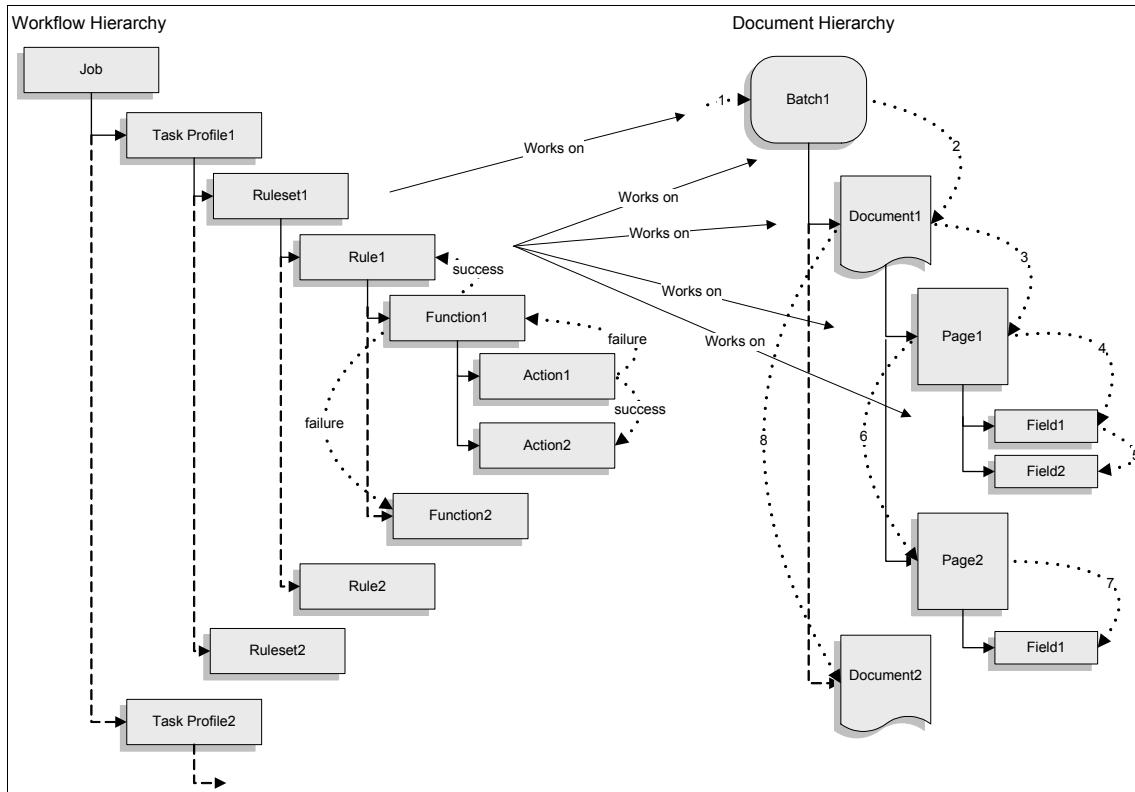


Figure 3-20 Workflow and document hierarchies and processing sequence

3.5.6 Function and action

A rule is made up of one or more functions. A *function* consists of one or more actions. An *action* represents the code that runs a particular operation on the objects of a document. A function is started in the order in which it appears in the rule. If an action fails, the function that called it exits unsuccessfully, and the next function in the sequence gets executed. If the action succeeds, the next action in the function gets executed. If all actions of a function execute successfully, the rule that called the function exits successfully. By using this approach, you can construct efficient processing rules without coding.

For example, in a rule to identify a type of page (“Page identification” rule), several functions can be assembled in a fallback sequence, from the most to the least processing intensive or efficient. Each function implements a specific recognition technology.

The rule calls the following functions:

- ▶ “Identify using fingerprint” function
- ▶ “Identify using text match” function
- ▶ “Identify using pattern match” function
- ▶ “Identify manually” function

Manual identification, in effect only flagging the page for a subsequent user-attended task, is called only after fingerprint, text, and pattern matching all fail. If the fingerprint matching function succeeds, that is, if all the actions in it succeed, the “Page identification rule” exits successfully.

For information about document type identification, see 3.6.6, “Classification” on page 86.

3.6 Taskmaster action libraries

Many features in Taskmaster applications are implemented by using the rich set of action libraries that are available in the product. They are used in the course of processing a batch and its documents. They are also used for export, lookups, integrating with other systems, and system reporting. This section provides a cursory view of these actions before looking at how they are used in the Taskmaster process in the next section.

The action libraries delivered with the base Taskmaster product cover the full spectrum of functionality required in the Taskmaster process. However, by further combining these actions into functions and rule sets, you can implement a much larger functional scope than suggested purely by the elemental actions. The Taskmaster Accounts Payable Capture and Medical Claims Capture add-on applications are examples of this approach. You might want to consider these applications if you intend to implement similar solutions.

Important: The use of several action libraries requires additional licensing that is not covered in the license for Production Imaging Edition. See Table 3-1 on page 77 for more information about these libraries.

You can use the actions in Table 3-1 as part of the base license for IBM Production Imaging Edition.

Table 3-1 Summary of Taskmaster actions with the base license of Production Imaging Edition

Actions (library name)	Functionality
Virtual scanning (VScan)	Creates a batch and imports existing images in it from a directory in the file system.
Batch splitting (Split)	Splits a batch into smaller batches that are processed separately.
Color to B&W conversion (ColorToBW)	Changes the color depth of images.
Grayscale to B&W conversion (Grayscale)	Converts grayscale TIFF to bi-tonal TIFF.
TIFF-PDF conversion (DCPDF)	Converts PDF to TIFF and TIFF to PDF.
Image conversion (ImageConvert)	Appends images in a single file, and converts BMP, GIF, and PNG file formats to JPEG or TIFFs and TIFFs to JPEG or vice versa.
TIFF file merging (TifMerge)	Combines individual TIFF images into a multipage TIFF file.
Image clean-up and enhancement (DCImageFix)	Provides filters to despeckle, deskew, remove color, enhance characters, and so on.
Clip images (Dcclip)	Clips a portion of an image and saves it as a separate TIFF file.
Imprinting and redaction (Imprint)	Provides capabilities to overlay text on, or redact part of, an image.
Image overlay (Ioverlay)	Combines the current page image with a background image.
Barcoding (Barcode_x, Barcode_p)	Locates and recognizes 1D and 2D barcodes, and tests barcode values.
ICR processing (ICR_C)	Recognizes hand-written block characters by using the OpenText Recostar engine.
OCR processing (OCR_A, OCR_S, OCR_SR)	Recognizes machine-generated characters by using ABBYY OCR engines, Nuance OCR engines, or a combination for voting purposes. The ABBYY engine can also be used for recognition of certain types of marks (OMR).
Recognition (Recog_Share)	Performs various fingerprint- and recognition-related functions, including recognizing bubble options (OMR) using pixel threshold evaluation.

Actions (library name)	Functionality
Runtime Document Hierarchy (DCO)	Sets up, tests, and modifies runtime information associated with each object of the document hierarchy in a batch (batch, document, page, and field). This information can include object statuses, types, recognition confidence levels, object variables, output field values, and fingerprint identifier.
Runtime zones (Zones)	Provides the capability to read and modify, at run time, zone information associated with each field in a fingerprint. It also provides the capability to locate the recognized text of specific zones, assign values to fields, and so on.
Fingerprinting (Autodoc and FingerprintMaintenance)	Provides capabilities to create and maintain fingerprints, match pages to them, and manipulate fingerprint information. Fingerprints are used to automatically classify structured and semistructured documents by using image analysis and field positional information.
Wordfire content-based classification (ICM)	Classifies text-intensive, free-form documents by analyzing their contents, rather than by relying on keyword, physical, or positional information. Requires the IBM Classification Module.
Data validation (Validations)	Extensive library to check and modify the content and format of field values. Also performs arithmetic calculations, assigns and copies values, checks variables, and so on.
Picture string matching (Picture)	Performs field validations by using <i>picture strings</i> that define the permitted character format of a field.
Data entry voting (Vote)	Compares the data of two data entry passes.
Intellocate	Updates the existing field position information in the document hierarchy or adds position information for a new fingerprint.
Locate text and navigate (Locate)	An extensive library used in combination with full text recognition to locate words or regular expressions on a page and to navigate around the page by line or word.
Lookup (Lookup)	Provides the capability to validate field values, by using database lookups, and to populate fields with lookup results.
Pattern matching (PatternMatch)	Provides the capability to search an image for a match to a pattern that is defined in the fingerprint library, for page identification and page registration (alignment).
File system operations (FileIO)	Provides access to the file system to copy, test, rename, and delete files and to set file attributes.
Output to email (Email)	Used to compose and send email from the Taskmaster Client by using CDOSYS (web) or a Simple Mail Transfer Protocol (SMTP) server (requires the Microsoft Outlook user to be logged in).

Actions (library name)	Functionality
Export to text file (Export)	Exports and formats document hierarchy information, such as object variables and extracted data fields, to a flat file that can be used to feed external systems.
Export to database (ExportDB)	Exports and formats document hierarchy information, such as object variables and extracted data fields, to a database through an ODBC connection.
Export to XML (ExportXML)	Exports and formats document hierarchy information, such as object variables and extracted data fields, to an XML file that can be used to feed external systems.
Export to FileNet Content Manager (FileNetP8)	Exports document contents and metadata to a FileNet Content Manager repository.
Export to FileNet Image Services (FileNetIDM)	Exports document contents and metadata to a FileNet Image Services repository.
Export to IBM Content Manager (ibmcm)	Exports document contents and metadata to an IBM Content Manager repository.
NENU utility actions (NENU)	These actions are used by the notification utility in Taskmaster for batch monitoring, status notification, and automatic deletion of completed batches.
Rulerunner utility actions (Rrunner)	Performs miscellaneous utility functions, including checking batch integrity, manipulating the values of fields and variables, raising condition flags, and controlling rule execution.

The actions in Table 3-2 require additional licensing.

Table 3-2 Summary of Taskmaster actions requiring additional licensing

Actions (library name)	Functionality
Input from Email (Iemail)	Imports image file attachments from an Exchange (through EWS) or Internet Message Access Protocol (IMAP) server into the current batch.
Electronic document conversion (Convert)	Converts Microsoft Excel, Word, Outlook email and attachments, image files, PDF to TIFF, and multipage TIFF to single-page TIFF.
Input from Fax server (OpenTextFaxServer)	Imports faxes from the inboxes of an OpenText Fax Server.
Export to SharePoint (SPExport)	Exports document contents and metadata to a Microsoft SharePoint repository.
Export to Documentum (Documentum)	Exports document contents and metadata to an EMC Documentum repository.

Actions (library name)	Functionality
Export to LiveLink (LiveLink)	Exports document contents and metadata to an OpenText LiveLink repository.

The following subsections expand on several key capabilities that are implemented in the action libraries and explain how they are used to get the job done.

3.6.1 Image cleanup and enhancements

The image cleanup and enhancements library contains actions that clean up and enhance images to improve legibility and data capture processing in the later stages of the Taskmaster process. It provides filters and settings to perform the following actions:

- ▶ Deskew or straighten a crooked image to improve OCR performance and improve reading.
- ▶ Rotate an image typically by increments of 90 degrees when, for example, certain pages that display in landscape mode were scanned as part of a batch that was processed in portrait mode.
- ▶ Deshade to increase crispness and better reveal text on shaded areas and graphics.
- ▶ Dilate or erode an image to increase or decrease the thickness of the shapes in the image without changing their proportions and to make them easier to process. This action is especially useful for character recognition when characters appear thin, with discontinuities (an “l” looking like an “i”), or conversely, as a mass lacking details (an “i” looking like an “l”).
- ▶ Despeckle and reduce noise to remove random specks or background noise typically introduced when the scanner sensitivity threshold is too low in black and white, or by shaded backgrounds in forms.
- ▶ Smoothen characters to repair broken segments and smooth ragged edges as occurs when scanning dot-matrix printed documents.
- ▶ Inverse text to detect and reverse regions of the image that typically have white text on a black background, such as in table headings.
- ▶ Remove horizontal and vertical lines in high-density forms and tables to reduce clutter and enhance recognition of useful data.
- ▶ Remove the black borders that typically form on the edges of a black and white image that was scanned with high sensitivity settings.
- ▶ Remove streaks, vertical lines, or smears that are sometimes added to an image by the scanning process.

- Fill line breaks to repair broken lines, as in underlined text, for example, and to improve human legibility.
- Create an outline out of shapes on the image to sometimes improve legibility of a busy image by dropping unnecessary details.

These filters are typically used in combination and their proper mix and settings are derived from multiple trials and errors to arrive at the optimal result.

3.6.2 Barcode recognition

The barcode libraries are used to automatically locate and recognize one or several barcodes in an image. Barcodes are used to store data that can be detected with high accuracy, even on poor quality documents.

Many types of barcodes have been developed over the years to serve the needs of specific industries, such as retail, logistics, manufacturing, postal services, healthcare, airlines, and transportation in general. The choice of using a specific barcode in a Taskmaster application can be dictated by the barcode standard in use by the customer. Alternatively, you can decide the choice of barcode arbitrarily to satisfy the internal needs of the application. Such needs might be to identify a type of form, mark document boundaries in a batch, or reconcile incoming documents with a work item or other documents that belong together.

Taskmaster can detect 1D and 2D barcodes. One-dimensional barcodes (Figure 3-21) store a limited amount of data, typically a single alphanumeric string that can be used as a reference. They are coded by using a pattern of vertical lines of varying width read along the horizontal axis of the barcode.



Figure 3-21 One-dimensional barcode Code 39

Figure 3-21 uses the Code 39 barcode to code a claim number that can be used as a reference on outgoing correspondence to a customer to automate the claim identification process when the mail returns. Code 39 can store up to 43 alphanumeric characters.

Taskmaster supports the following one-dimensional barcodes: 2of5, Interleaved 2of5, Airline 2of5, Matrix, Matrix 2of5, Code 32, Code 39, Code 39 Extended, Codabar, Code 93, Code 93 Extended, Code 128, EAN13, EAN8, UPC-A, UPC-E, Addon5, Addon2, UCC128/EAN128, Patch Code, and PostNet.

Two-dimensional barcodes (Figure 3-22) can store up to several kilobytes of data. They are coded by using a matrix that represents information along the vertical and horizontal axes of the barcode.

The PDF417 barcode is a popular two-dimensional barcode that has many uses including to code driver license information in certain states. It codes information in multiple rows (from 3 to 90) that show clusters of bars and spaces. It is described as a “portable data file”, because the barcode itself carries the information, not just a reference to it.

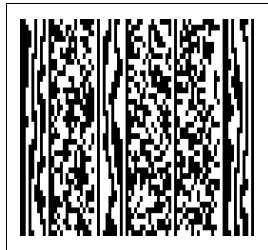


Figure 3-22 Two-dimensional barcode PDF417

Figure 3-22 codes the entire postal address: “International Business Machines, 3565 Harbor Boulevard, Costa Mesa, California, 92626-1405, United States.”

Taskmaster supports the PDF417 and Datamatrix barcodes.

3.6.3 Optical Character Recognition

OCR technology is used to convert machine-printed text in an image back to editable text. It is at the core of the Taskmaster document identification and data capture process and is used for full page recognition and zonal data extraction. Taskmaster includes three OCR engines that can be used individually or in combination to offer the best possible results.

Although implementations differ, the three engines operate in a similar fashion. At a high level, the OCR engine analyzes the image and the textual zones and isolates individual characters. It then compares each character to a collection of template character bitmaps (in various fonts) and selects the closest match. It assigns each character a “recognition confidence level” based on how well it correlates with the template. Then it assembles the characters into words and resolves ambiguities by using dictionaries or lexicons and various other techniques.

The confidence levels are measured against the confidence thresholds set in Taskmaster. The higher the OCR confidence threshold is, the higher the number

of errors is. Confidence levels of recognized zones are saved in the document hierarchy at run time to be used to color code the fields and drive the focus to problem fields in the Taskmaster Verify user interface. It is also possible to run multiple OCR engines on the same image to achieve the most accurate results across them all, which is a technique called *voting*. In that case, in each successive pass, Taskmaster compares the confidence level of the recognized data in the current pass with the one recorded in the previous pass. Then it only updates the field with the new data and the confidence level if it is higher.

Although the accuracy of OCR engines has improved over the years, it is affected by many factors. These factors include the page layout and background, image resolution, color/contrast/brightness, skewing, jagged characters, font type, size, and emphasis, type of compression, language, and character set. However, in most Taskmaster implementations, the following factors can be adjusted to produce the best possible results:

- ▶ Resolution, color, brightness, contrast, and skewing can be tuned at the scanner level or corrected by using Taskmaster image cleanup and enhancement actions.
- ▶ Character attributes, such as jagging, thickness, or thinness, can be corrected again by using image enhancement actions.
- ▶ Page layout, font type, and font size can be adjusted if the organization can influence the design of forms.
- ▶ Uncompressed or CITT Group 3/4 TIFF can yield better recognition results. In such case, Taskmaster offers the capabilities to convert to TIFF incoming images with other formats and compression schemes. They can still be carried through the Taskmaster workflow by using a separate stream to the repository. Alternatively Taskmaster can convert the TIFF images to a more compressed format at the Export stage.
- ▶ Recognition performance of national languages and character sets (especially accented characters) can be adjusted by selecting the OCR engine that yields the best results or by using voting.

3.6.4 Intelligent Character Recognition

ICR technology converts hand-written characters in an image to editable text. Figure 3-23 shows ICR technology used in a sample application.

The screenshot displays a tax form application with handwritten data processed by ICR. The form is divided into several sections:

- Personal Information:** Includes fields for "Your first name and initial" (John), "Last name" (Doe), "If a joint return spouse's first name and initial" (Jan), and "Last name" (Doe). The home address is listed as "4127 Crestridge, Columbia, CA".
- Identification:** Fields for "on file" and "See" are present.
- Financial Data:** A series of boxes contain handwritten numbers and their corresponding ICR results:
 - Box 1: 12 950 00 → 1295000
 - Box 2: 18 640 21 → 1864021
 - Box 3: 2 279 14 → 227914
 - Box 4: 0 → 0
 - Box 5: 2 279 14 → 227914
 - Box 6: 1 774 00 → 177400
- Summary:** A list of items on the left includes "John Doe", "Jan Doe", and "4127 Crestridge".

Figure 3-23 ICR in a Taskmaster sample application

The overall operating principle of ICR is similar to OCR. However, because of the variability in handwriting, the techniques to separate and classify characters are different. Instead of trying to isolate and match whole characters, individual handwriting strokes are isolated and analyzed spatially to see which strokes belong most likely to which characters. Also, character classification requires a much broader base of shapes and more complex methods, including statistical probabilities, to disambiguate and identify characters and words with confidence.

ICR is affected by the same factors as OCR. However, it is affected by the clutter introduced by handwriting going over the boxes used in form fields and for normalizing character spacing ("constraint boxes"). Line removal or repair and dropout are typically employed to make the text stand out and improve accuracy.

3.6.5 Optical Mark Recognition

OMR is used in Taskmaster to detect whether check boxes, bubbles, or other types of marks have been selected. It is typically used in combination with other Taskmaster functionality that applies processing logic to the basic OMR results to interpret and turn them into actionable data. The results depend on the purpose and type of prompt or answer expected. For example, the answer might be yes or no, yes or no to all that apply, multiple choice, grid to form numbers or words, or add up marks.

Figure 3-24 shows OMR technology used in a sample application.

mark one answer by putting an "X" in the appropriate box [X] for each statement.)

	Excellent	Very Good	Good	Fair	Poor	Don't know
a. Providing products and services to meet the needs of your business	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
b. Having rules and regulations that are easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
c. Having employees who interpret rules and regulations consistently	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

1a: Providing products and services to meet the needs of your business. Fair

1b: Having rules and regulations that are easy to understand. Good

1c: Having employees who interpret rules and regulations consistently. Good

1d: Providing products and services that are good value for the price. Good

Figure 3-24 OMR in a Taskmaster Survey sample application

Although this technology has been in use for years, it is not easy to address the many situations that can arise and to determine the confidence levels and errors that trigger the Verify task. Factors that affect accuracy include the type of marks, filling method, variability in the response of the filler (spill over, too light, erasure), and interfering specks and background noise in the image.

Taskmaster provides flexibility to address these cases. When configuring the zones and fields, you define a parent field (for example, "seat") as an OMR group and subfields (for example, "window" and "aisle") to host options that belong together. You also specify in the parent field the number of options and whether multiple can be selected.

To do the actual recognition, Taskmaster offers two methods to address various situations. In most cases, when regular check boxes are used, the quickest setup is to use the OCR_A (ABBYY) engine. When the outline of the check box becomes removed through line removal or dropout, which is sometimes necessary to process high density forms, or when using bubbles or unusual check marks, you might need to use the "pixel evaluation method" instead. In this method, the zone defined for the check box is evaluated against a threshold of darkness (black pixel percentage) and a threshold of background noise that you specify to determine what is considered a checked mark. The difficulty with this technique is that it is more sensitive to variations in background noise or specks, which affect the pixel count for the zone. Therefore, adjustments might be necessary to find the appropriate values of these thresholds.

3.6.6 Classification

A great return on investment is derived from the ability of Taskmaster to automatically classify documents and drive the data extraction and indexing process when exporting the documents to the back-end repository. Therefore, it is one of the most critical pieces of functionality.

Given the variability in the types and quality of the documents that are routinely processed, it is often necessary to use a combination of several techniques to reliably identify documents. Taskmaster the following ways, among others, to classify documents:

- ▶ Manual page identification
- ▶ Structure-based identification
- ▶ Text and pattern matching
- ▶ Barcodes (see 3.6.2, “Barcode recognition” on page 81)
- ▶ Fingerprinting (see 3.6.7, “Fingerprinting” on page 87)
- ▶ Content-based identification with IBM Classification Module (see 3.6.8, “Content-based identification with IBM Classification Module” on page 87)

Manual page identification

Manual page identification is used as a last resort when every other method has failed. With the help of automatic fingerprinting or content-based classification with IBM Classification Module, the number of manual interventions should decrease over time.

Structure-based identification

Structure-based identification is used in cases when the batch is fairly structured, that is, when the succession of pages can be predicted and used to determine the document type. In such case, you can use the actions of the runtime document hierarchy to arbitrarily set the page types.

Text and pattern matching

Text and pattern matching techniques are used on structured and semistructured documents. They are used when the types of documents are so close and the relative positions of zones are constantly changing that fingerprinting is unable to detect the type accurately.

Instead of looking at the overall page, as fingerprinting does, text matching attempts to identify a document based on searching keywords and phrases that unequivocally determine the type of document. It is called after the document has run through recognition, and therefore, typically requires more processing than just image analysis. Pattern matching concentrates on specific graphical marks or anchors in areas of an image.

3.6.7 Fingerprinting

The most innovative Taskmaster feature in the area of document classification is the use of *fingerprints*. A *fingerprint* is a unique signature of a page that is saved in the system and used to automatically classify incoming documents against those documents that have been processed before. The idea is that, if the fingerprint of the incoming document matches an existing one, you can safely assume that the incoming document is of the same class as the existing one. This technique is particularly well adapted for structured and semistructured documents that exhibit a fairly constant layout.

A fingerprint is made up of a sample image with a representative layout of the class of document and information that describes its geometric profile based on analyzing its pixel distribution. It can also be complemented with recognition results. The fingerprint is assigned a unique identifier that is saved in the fingerprint database.

Although all the documents of a same class look alike from a distance, every instance of a document is typically unique in that its actual contents are different from the ones before. Therefore, the chances of detecting the exact document are highest when the incoming document is a copy of the original document. Detecting an instance of an already identified class is a matter of measuring the proximity of the fingerprint of the incoming documents to the existing ones. The closest match has the highest probability of the instance belonging to the identified document class.

Fingerprints get perfected over time with additional information when more instances of the same class are routed to be identified by operators in the Taskmaster process. By using the fingerprinting and *Intellocate* libraries, you can configure your application to automatically create a new fingerprint and zone positions after an unrecognized page has been routed to a Verify task. This way, only verified recognition results are saved in the fingerprint, enhancing accuracy.

3.6.8 Content-based identification with IBM Classification Module

The techniques mentioned previously all have in common that they look for specific features of the document to identify and separate it from others in the batch. However, they work less well in cases when you need to process a mix of mostly text documents. Examples include miscellaneous customer correspondence, complaint letters, policies, statements, or affidavits with no predictable structure, logo, barcode, marks, or keywords. In such cases, you must understand the content in the same way as a person who is unable to recognize a type of document at first glance needs to read its content to make the determination. For this reason, Taskmaster relies on IBM Classification Module.

Important: IBM Classification Module connectivity is part of the base Taskmaster product and is included in the license for IBM Production Imaging Edition. However, IBM Classification Module connectivity depends on Classification Module, which requires a specific licensing.

Similarly to fingerprinting, IBM Classification Module creates a unique identity exclusively from the textual contents of documents. It looks for patterns, concepts, and associations and stores the results mathematically. This identity is then associated with a given type of document. Initially, document identification requires human intervention to match a given identity to a document type. However, IBM Classification Module can learn from the processing of a range of sample documents, and over time it requires no manual intervention.

At run time, the IBM Classification Module connector invokes IBM Classification Module and passes to it full-text recognition results. IBM Classification Module analyzes the content and compares it to its collections of identified types of documents. If it finds a match, it returns the type to Taskmaster. Otherwise, Taskmasters assigns a low confidence rating to the document, which causes it to be classified by an operator.

Because IBM Classification Module analyzes documents in their entirety based on concepts, it has a much larger scope to accurately identify a document than other methods can do, based strictly on a linguistic approach. Also the internal representation of information in IBM Classification Module makes it fairly immune to OCR or ICR and manual input errors.

In summary, with the help of ICM, you can perform the following tasks:

- ▶ Automatically identify text-intensive, free-form documents.
- ▶ Reduce prescan manual sorting and document separating.
- ▶ Enable automatic processing of mixed document batches.
- ▶ Process noisy OCR or ICR documents without operator intervention.

3.6.9 Language support

Taskmaster supports the several languages in the following ways:

- ▶ GUI: Brazilian Portuguese, Dutch, English, French, German, Italian, Polish, Spanish, and Swedish
- ▶ Data entry, display, and internal processing of character sets: All Latin 1 to 4 character sets
- ▶ OCR/A (ABBYY) engine: Czech, Dutch, Dutch Belgian, English, Finnish, French, German, Hungarian, Italian, Lithuanian, Polish, Romanian, Slovak, Spanish, and Turkish

- ▶ OCR/S (Nuance) engine: Nuance Omnipage (OCR/S, ocr_s and ocr_sr libraries): Afrikaans, Albanian, Catalan, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, German, Hungarian, Italian, Icelandic, Latvian, Lithuanian, Maltese, Norwegian, Polish, Portuguese, Slovakian, Spanish, Slovenian, Swedish, Turkish, Esperanto, Romanian, Serbian (Latin), Faroese, Gaelic Irish, Gaelish Scottish, Rhaetic, Sami, Northern Sami, Southern Sami, and Swahili
- ▶ ICR/C (RecoStar) engine: Afrikaans, Albanian, Bosnian-Latin, Catalan, Croatian, Czech, Danish, Dutch, English, Estonian, Faroese, Finnish, French, German, Hungarian, Icelandic, Irish, Italian, Latvian, Lithuanian, Norwegian, Polish, Portuguese, Rhaeto-Romanic, Romanian, Serbian-Latin, Slovak, Slovenian, Spanish, Swahili, Swedish, and Turkish
- ▶ IBM Classification Module: Dutch, English, French, German, Italian, Norwegian, Portuguese, Spanish, and Swedish

In addition, the following vertical dictionaries can be selected in the following languages for the OCR/S engine:

- ▶ Legal and medical dictionaries: Dutch, English, French, and German
- ▶ Financial dictionary: English

3.6.10 Imprinting and redaction

Taskmaster provides an imprint library that can be used to overlay text on an image. Alternatively, the library can be used to redact part of the image to protect personal information from public view, which is the case with health records and social security information.

You call the redaction action in a rule, attached to a target field defined in a fingerprint, to black or white it out. Alternatively, you can use text locating actions to find, navigate, and position automatically the redaction zone over the data in unstructured documents. The resulting imprinted content or redaction is flattened and burned in the image.

Typically when redaction is used, two capture streams are implemented in Taskmaster. First, the redacted documents are committed to a repository with access rights for general circulation. Then the originals are committed to a secure repository with restricted access rights for authorized personnel.

3.6.11 Locating text

Taskmaster offers an extensive library of actions that can be used in combination with text recognition to locate words and regular expressions and to navigate around a page. These actions are used in instances when text location cannot be predicted, such as in semi-structured or free forms. They are coupled with actions such as those used to process line items in purchase orders and invoices.

3.6.12 Validations

Taskmaster offers an extensive library of elemental actions. These actions help you to validate and manipulate the data captured in the objects of the runtime document hierarchy and to ensure that they conform to your business rules.

The following actions are possible:

- ▶ Data formatting actions to normalize field values or prepare them for calculation or comparison purposes, including the following actions:
 - Padding with zeros or spaces to match the expected number of characters
 - Deleting a specific character in a specific position or all instances
 - Deleting a class of characters from a field (alpha, numeric, punctuation, nonalphanumeric, or system characters)
 - Testing data types (date, currency, alpha, or numeric) and field length
 - Converting the case of characters or value to currency
 - Clearing a field value
 - Inserting a decimal point or a specified character in an existing field value
 - Trimming spaces and truncating and splitting field values
 - Parsing postal addresses and names and populating individual fields
- ▶ Manipulation of field values and document hierarchy variables, including the following actions:
 - Assigning default values to fields
 - Copying or appending values between fields
 - Comparing field values and verifying arithmetic calculations between fields
 - Comparing dates and testing them within a range or days
 - Assigning data or a time stamp to a field
 - Testing field content: Filled, empty, max or min length, specified matching value, or percentage numeric or non-numeric data
 - Testing the number of OMR boxes and whether they are selected

- Testing a regular expression in a field
- Testing variables and assigning variables to fields
- Summing up values of subfields
- ▶ Invoking a message box to provide guidance in the Verify task

For example, by using combinations of these actions, you can create rules and attach them to any object of the document hierarchy to test the following information:

- ▶ Values are within accepted ranges.
- ▶ Data is formatted as required.
- ▶ Dates are valid, and deadlines are met.
- ▶ Numbers add up or are not missing.
- ▶ Mandatory fields and check boxes have been completed.
- ▶ Dependencies between data are respected.
- ▶ Data matches sets of permitted values, by combining with lookup actions

Upon failure of any of these rules, Taskmaster flags the associated field and page for manual review in a Verify task, similarly to low confidence recognition results.

Lookups

As a good practice, check and normalize data early in the business process to reduce errors and enforce consistency. To achieve this objective, Taskmaster provides a set of actions that you can use to connect through ODBC or OLE DB to a business database hosting reference information. Such information might include customer names, part numbers, or nomenclature. Then you run an SQL query and pass back a result set to populate fields in the runtime document hierarchy.

By using the rule execution logic outlined in “Rule processing” on page 71, you can handle lookup errors through a function that is called upon failure of any of the lookup functions.

3.6.13 Exports

The Taskmaster process completes the processing of a batch by persisting the captured documents and metadata to the Enterprise Content Management (ECM) repository. In many cases, it also exports some of the business data to line of business systems such as databases or applications. It can output data and contents in a format that is compatible to the input stage of another system, such as IBM Content Manager On Demand.

Taskmaster includes libraries to export to IBM and non-IBM ECM repositories, relational databases, XML, and flat files.

Exporting to FileNet Content Manager

In a Production Imaging Edition implementation, you use the FileNet P8 action library to commit documents to FileNet Content Manager. It provides the following capabilities:

- ▶ Establish a connection to a FileNet Content Manager system.
- ▶ Attach to a given object store and FileNet P8 document class.
- ▶ Define a root folder and create a subfolder to store your documents.
- ▶ Map the Taskmaster field values and variables of the runtime document hierarchy to the properties defined in the FileNet Content Manager document class.
- ▶ Upload documents to the destination folder.

To bind preparation actions, such as connect, login, and create directory, to the open event at the batch level, bind the populating of document properties. Then upload to the open event at the document level, which automatically iterates for each document of the batch. Figure 3-25 shows the actions to export documents to FileNet Content Manager.

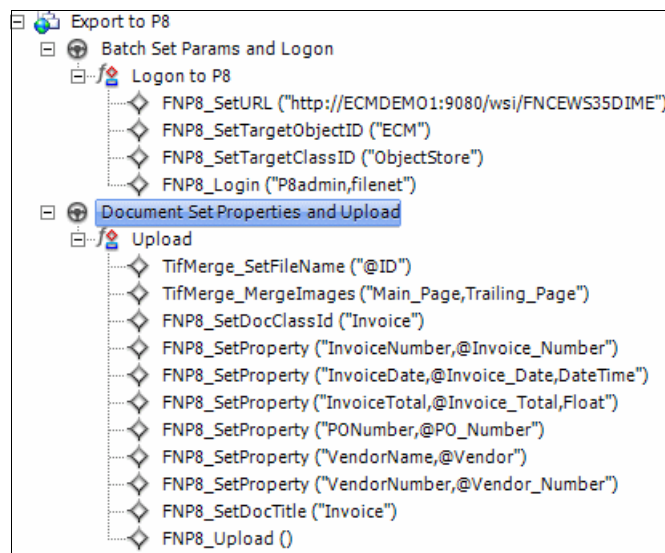


Figure 3-25 Actions to export to FileNet Content Manager in Datacap Studio

Exporting to a database

Similarly, to export to a database that is accessible through ODBC or OLE DB (Figure 3-26), the ExportDB library provides the following actions:

- ▶ Establish and close a connection to the database.
- ▶ Open the target database table.
- ▶ Assemble each database record in memory, and populate it with data from Taskmaster field values and variables of the runtime document hierarchy.
- ▶ Commit the database record.

You bind the actions to open the database to the open the events at the batch level. You also bind the actions to close the database to the close events at the batch level. In addition, you bind the actions to create the data records to the open event at the document level.

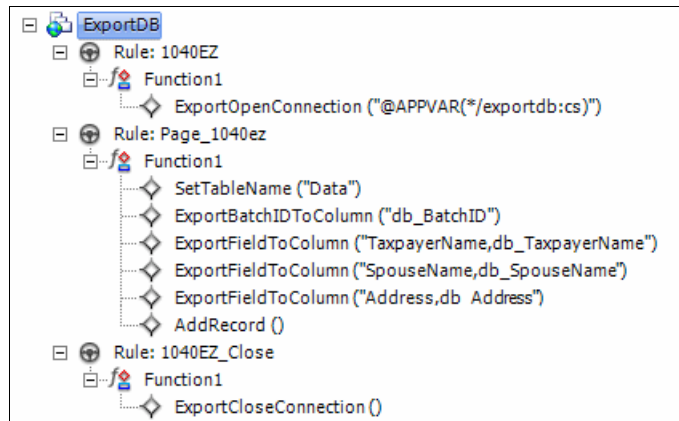


Figure 3-26 Actions to export to a database in Datacap Studio

Exporting to a flat file

By using the Export library, you can output data to the file system to be picked up for import by another system. The library provides the following actions:

- ▶ Set the path, file name, and extension of the export file.
- ▶ Format text (new line, blank lines, fields, filler characters, value and OMR separators, value justification, value length, OMR separator, and so on)
- ▶ Output text, date, time, field values, and variables of the runtime document hierarchy, and filter on field status
- ▶ Save the export file

The information that needs to be exported and how to code the export file depend on the target system. For example, this method can be used to feed the

Content Engine Bulk Import Tool as an offline alternative to a direct connection to FileNet Content Manager. Alternatively, it can be used to import documents in IBM Content Manager On Demand by using its ARSLOAD utility.

You bind the actions to create the export file and write out the data needed once to the open event at the batch level. Then you bind the actions to output the information required for each document to the open event at the document level. Finally, you bind the action to save the file to the close event of the batch level.

3.7 Principles and tools of the Taskmaster configuration

This section reviews the tools that are available to configure, deploy, and monitor a Taskmaster application, and to report on its activities.

At a high level, the principles for setting up a Taskmaster application are easily understood if you visualize a document and the types of data that you are trying to extract from it. A document is made up of pages that are typically identifiable by certain characteristics. Such characteristics include a specific structure (cover and trailing pages), the layout of each page, and the location in the page of the specific pieces of information that you are seeking to extract.

Configuring an application consists of providing Taskmaster with a combination of visual clues and processing rules from its catalog of actions. They drive how to automatically recognize and separate the documents; find, capture, and process the data in them; and transfer both images and data to the back-end systems.

Figure 3-27 shows a Taskmaster application and various areas of configuration.

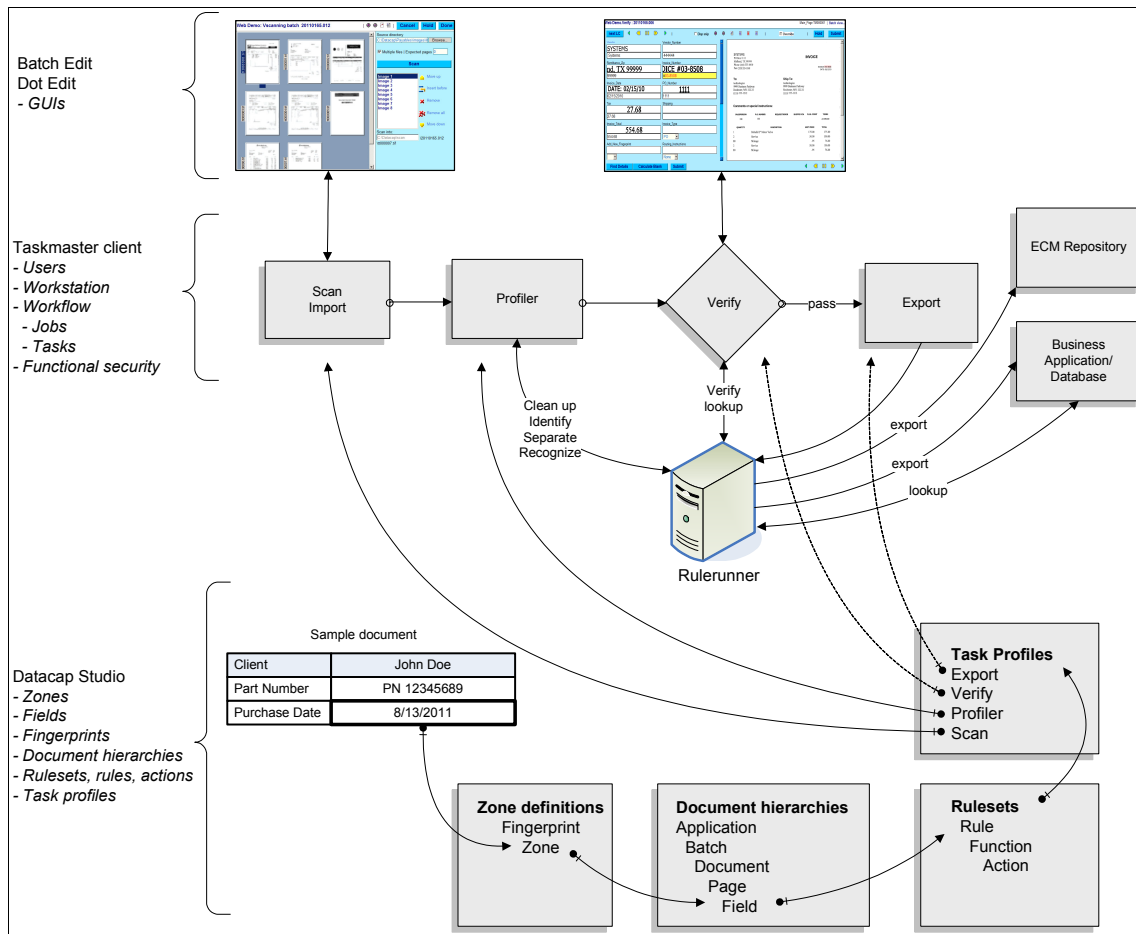


Figure 3-27 Configuring a Taskmaster application

You use *Datacap Studio* to configure the following information:

- ▶ The document hierarchy for your document types
- ▶ The fingerprints based on your sample document types
- ▶ The zones and data fields to extract
- ▶ The rules and actions to execute on the objects of the document hierarchy
- ▶ The tasks profiles to expose as actionable functionality in the workflow

For user-attended tasks, you use *Batch Pilot* or *Dot Edit* to define and bind the user interface specific to the current task. For more information, see “Batch Pilot for user-attended tasks” on page 62.

After the task profiles and core functionality of the application are designed and tested in Datacap Studio, you expose them to users, workstations, and Rulerunner by using *Taskmaster Client*. For more information, see “Taskmaster Client (thick client)” on page 58.

If you implement a Flexible Capture application, you must use *Flex Manager* to configure the data structures and document classes that you expect in your application.

Use the *Taskmaster Application Manager* to deploy your application on multiple machines and manage the registry of the Taskmaster system components. Use the *RV2 Report Viewer* to configure and access Taskmaster real-time activity reports over the web. Use *NENU* to set up and manage system health monitoring and notification, in addition to recurring house cleaning tasks.

3.7.1 Datacap Studio

Datacap Studio is the primary tool to configure and test Taskmaster applications. It offers an intuitive visual environment that makes it easy to assemble the various components of an application. It is organized around three work areas:

- ▶ Rule management for defining the document hierarchy to the field level and for tying processing operations to its components
- ▶ Zone management for creating classes of fingerprints from document samples and for defining graphical extraction zones tied to the fields of the document hierarchy
- ▶ Testing for running batches, checking the state of the objects in them, and debugging your application

Everything in Taskmaster is defined within the scope of an application, including the document hierarchy, zones, fields, fingerprints, rule sets, and tasks. The workflow and allocation of tasks to users and workstations and the user interfaces that are executed by user-attended tasks are also defined.

An application focuses on a specific set of document classes. The more document classes there are to process, the more complex the application is because more functionality must be built into the application to discriminate and address the various cases.

The document hierarchy describes the structure of the documents that are processed in the application, including batch, document, page, and field. This information is the basis for most of the other setup activities, because it describes the objects that are the targets of subsequent processing.

To create the document hierarchy, you import in Datacap Studio template pages that are representative of the class of document. For each page, Taskmaster creates a specific identity, called a *fingerprint*, based on its visual characteristics (Figure 3-28). Then you further define the zones where you want Taskmaster to extract information from. Taskmaster uses this information to automatically determine the type of document and identify the fields where to extract data.

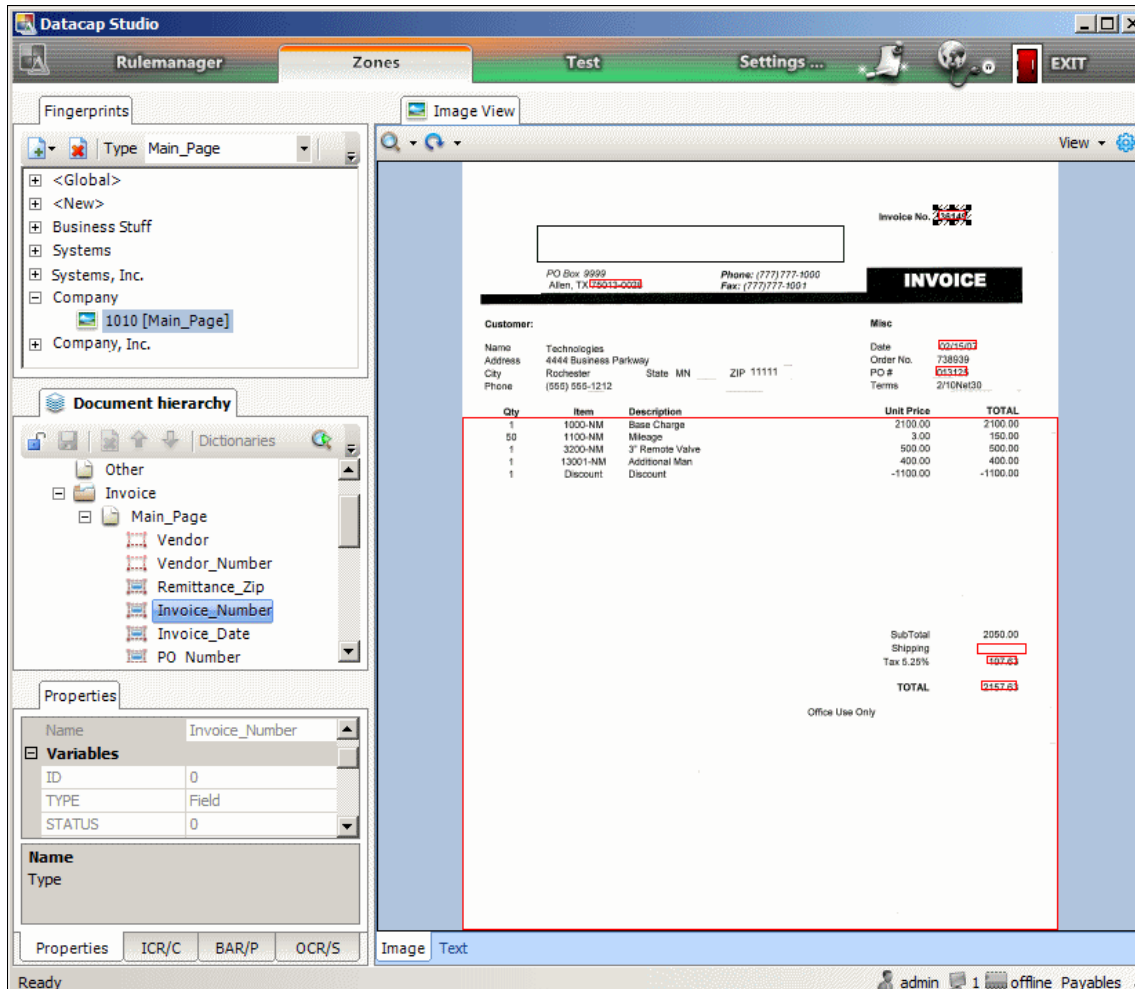


Figure 3-28 Defining fingerprints, zones, and data fields using Datacap Studio

Rulerunner runs rules and actions to perform the actual extraction work, document identification, data manipulation, and all the processing that happens behind the scene. You need to specify the actions that must be performed on which object of the document hierarchy for the particular task being executed. In

concrete terms, you must assemble elemental actions into functions and rules and bind them to the appropriate elements of the hierarchy.

For example, at the batch level, you need to configure rules that apply to the entirety of a batch when it is processed, such as when importing, scanning, or exporting. At the document level, you set the index properties for the whole document from the extracted data when exporting to a repository. At the page level, you set OCR or barcode options when performing recognition and data extraction. At the field level, you typically set field validation rules.

Rulerunner is readily available with hundreds of functions and actions that can be assembled in rule sets as needed. To expose and invoke the Taskmaster functionality in the workflow by operators or Rulerunner, you assemble the appropriate combinations of these rule sets as task profiles that can be used by the Taskmaster clients. Examples of the tasks include Scan, Profiler, or Verify. Figure 3-29 shows actions and rules mapping to the document hierarchy.

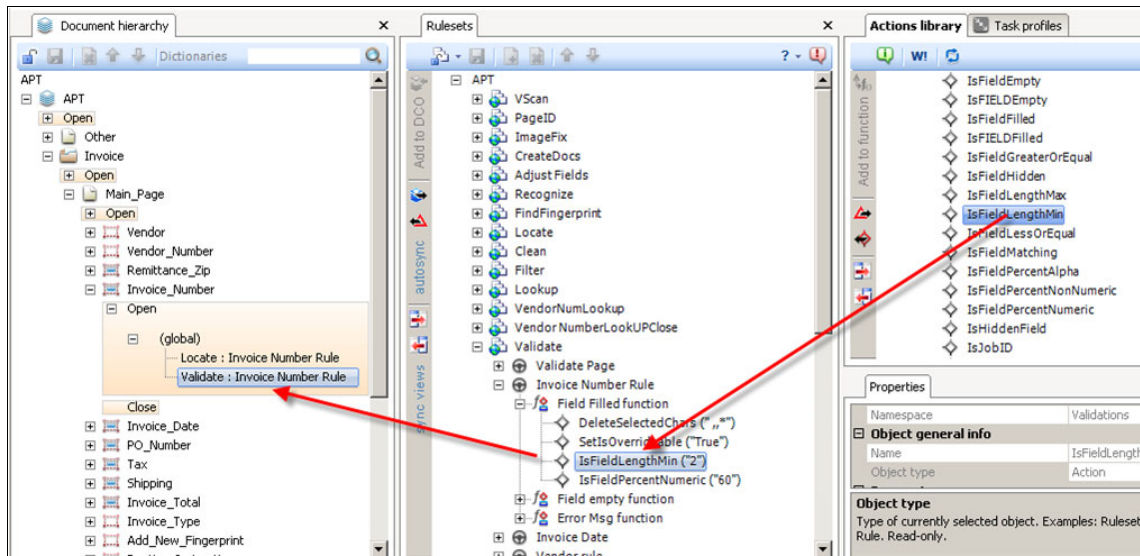


Figure 3-29 Datacap Studio - actions and rules mapped to document hierarchy

After you complete the configuration of your application, you can use Datacap Studio to test it. For example, at run time, you can test the values that are being extracted from the zones and check that the actions produce the expected results.

To help you start, Datacap Studio also includes an Application Wizard that generates the basic infrastructure of a new application. It takes you through the following steps:

- ▶ Creating a document hierarchy
- ▶ Creating a fingerprint class and template document
- ▶ Adding a few sample images
- ▶ Creating the application directory structure and configuration files for that application to run, including a standard workflow with thick and web client jobs.

With Application Wizard, you can also derive a new application from an existing one. This way, you can take any of the sample or foundation Taskmaster applications that most closely resemble your own use case. Then you can modify it to match your needs by using Datacap Studio, giving you a substantial head start.

3.7.2 Flex Capture and Flex Manager

The Flex Capture sample application is intended for cases where you can describe the types of documents that exhibit predictable types of data, if not a predictable layout. Therefore, by using its *Click'n'Key* and *Intellocate* functionality, Taskmaster quickly learns how to automatically recognize the documents from the input of an operator.

To set up the application, you define the data types that you expect in the documents, such as a purchase order number, social security number, or part number. You define the types of data that have a unique character pattern and that can be automatically located and used to help identify the type of document.

In most cases, you can expect to have much variability within a given class of document. For example, purchase orders are likely to all exhibit something called a “Purchase Order” with a specific numeric pattern. However, they might be found in different locations or in combination with a varying number of other types of data. As humans, we can easily recognize classes of document by looking at them and identifying the data types even though they are not always the same. The idea implemented in the Flex Capture application is to get Taskmaster to learn from a human operator by recording the moves of the operator.

When a document cannot be automatically recognized, it is flagged as a problem document and is routed to an operator for manual processing. The operator then selects the class of document and directs Taskmaster to automatically locate the expected data types for that particular class of document. However, it is likely that Taskmaster will be unable to locate some data because the document deviates from the known fingerprints. In such case, the operator needs to locate and select the missing data in the image for actions to be recorded by Taskmaster.

Then, when the batch is finally processed in the Export task (after having been verified), Taskmaster automatically adds the image and recognition information to the fingerprint collection. It also adds the location of the zones that have been selected to the document hierarchy. When the same type of document is processed again, Taskmaster automatically recognizes the document type.

Flex Manager is a utility that comes with the Flex Capture sample application. As shown in Figure 3-30, it helps you to define individual data types with the character filter to exclude unwanted characters and a picture string to specify the data format. It also helps you define whether the data is required and to define a regular expression to define the range of valid values.

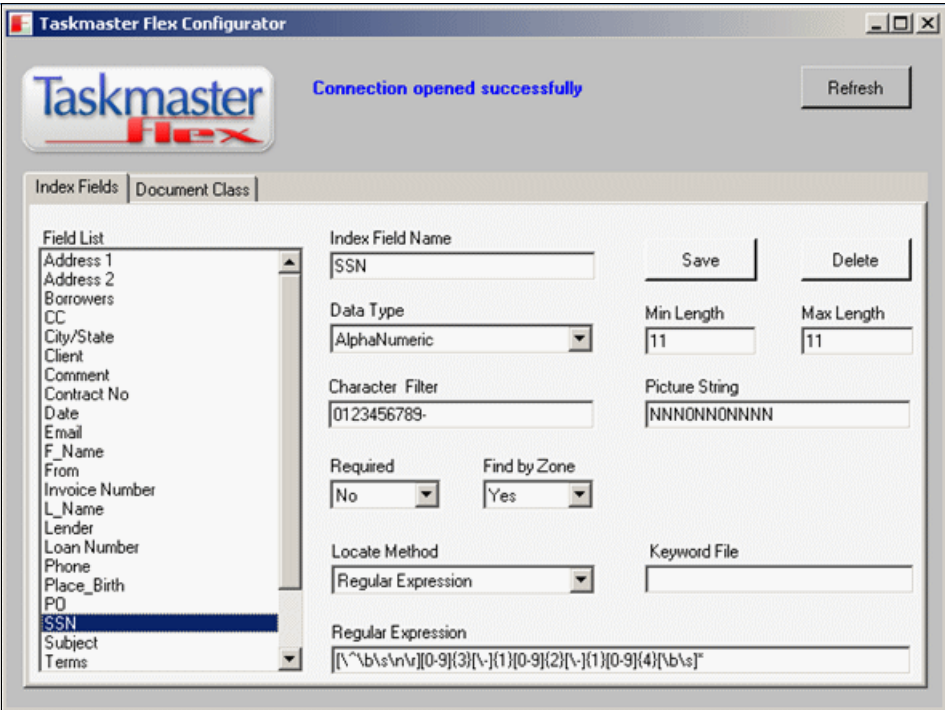


Figure 3-30 Defining the data types by using Flex Manager

After you define the data types that you need, you can create the document class that matches your type of document and assign it the appropriate combination of data types as shown in Figure 3-31.

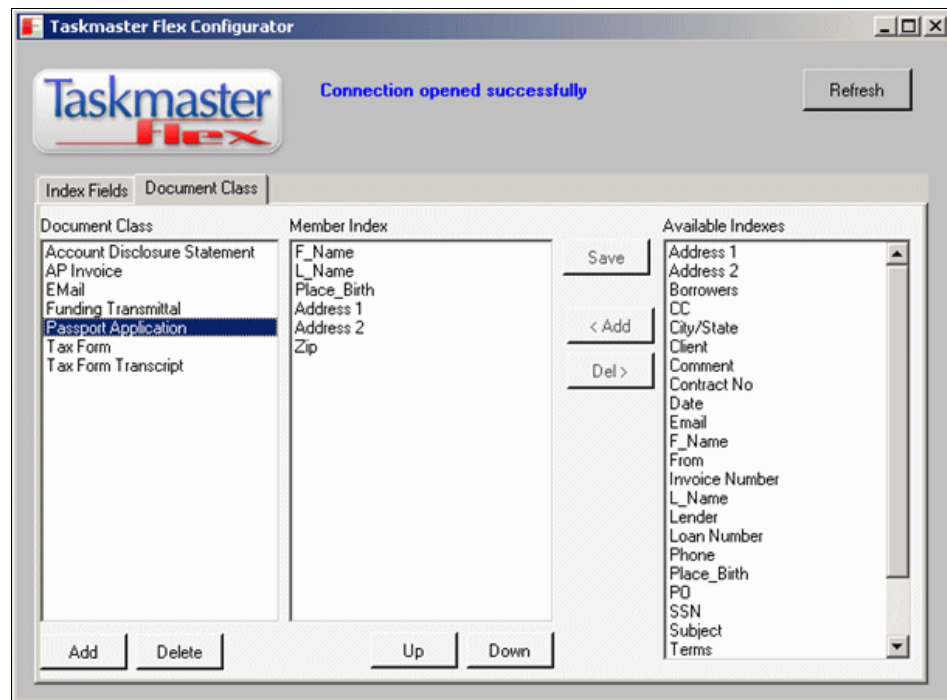


Figure 3-31 Defining document classes by using Flex Manager

3.7.3 Taskmaster Application Manager

Application Manager manages environment-specific information about the applications of a Taskmaster system in a central registry. This way, the Taskmaster system components that are running on different machines can be made aware of each other and get access to the information they need to operate. Such information includes pointers to the Taskmaster Server, databases, fingerprint library, and file server.

Application Manager maintains a separate registry for each application, which is cross-referenced by the central registry. To deploy an application on multiple machines, or from a development to a production system, you move the resources of the application (configuration files, working directories, databases, and so on) to their target machine. Then you update the registry of the application with the new locations and the cross-references between the registry of the application in the central registry.

Application Manager (as shown in Figure 3-32) is typically installed in restricted (read-only) mode on all workstations, except the machine running Datacap Studio to prevent users from modifying their deployment settings. In restricted mode, only the reference to the central registry can be modified.

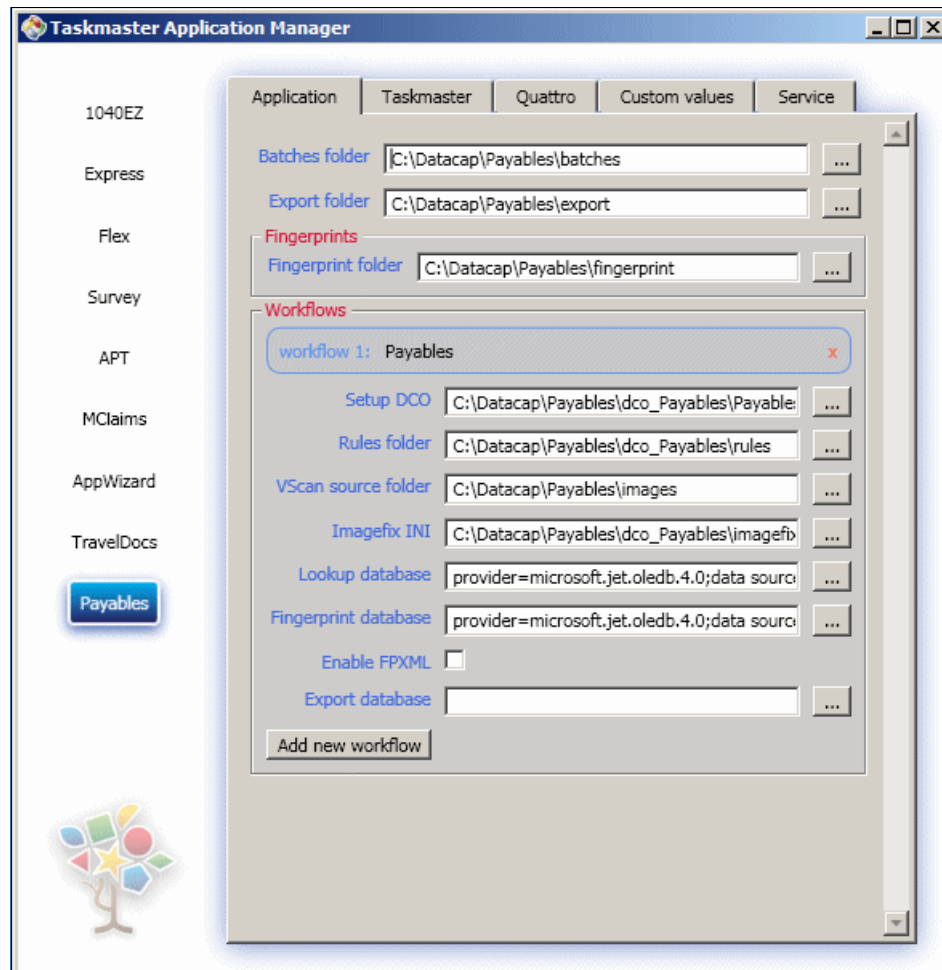


Figure 3-32 Taskmaster Application Manager

In Application Manager, you can define which tasks you want Rulerunner to run automatically in the background, for each application. You can also store application-specific custom values, such as a connection string to a lookup database or even credentials. These values can be retrieved and passed at run time to Rulerunner actions, avoiding hardcoding this information in the rules and making them more portable and secure.

3.7.4 RV2 report viewer

RV2 is a web-based tool to display real-time activity reports on the Taskmaster system. It is delivered with a set of preconfigured reports to monitor batch status, station activity, problem batches, batch aging, batch productivity, and so on, across multiple applications. Custom reports can be developed by using Microsoft Visual Studio and Windows Forms.

You can set up RV2 to report across multiple applications, display individual reports or a dashboard of reports, and filter on specific column values. Report contents can also be exported to PDF or Microsoft Excel, or they can be printed out. Figure 3-33 shows the RV2 report viewer.

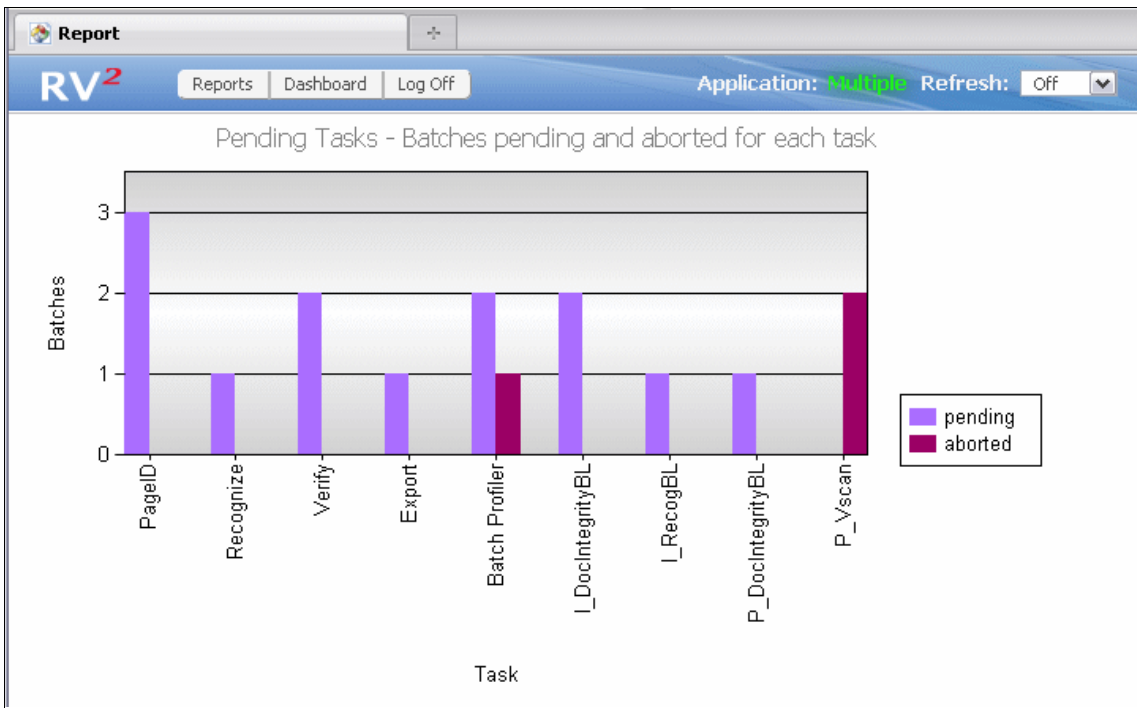


Figure 3-33 RV2 report viewer - pending tasks at various steps in the process

3.7.5 NENU

The NENU component automates recurring system health and house cleaning tasks, such as batch monitoring, status notification, and automatic deletion of completed batches. Tasks are scheduled by using the Microsoft Windows Scheduler.

NENU is a versatile tool that can execute any rule sets and actions defined for it in Datacap Studio by associating its rule set and task profile to the applications that you want to monitor. Typically you use NENU to perform selections in the Engine database of the application and execute actions on the selected batches. For example, you might run the following actions:

- ▶ Monitor batches, notify statuses, and automatically delete completed batches.
- ▶ Identify batches that meet certain criteria, such as batches that stopped.
- ▶ Change the status of batches and their order in the queue.
- ▶ Delete batches or move them to another location.
- ▶ Capture data snapshots to a database to be reported by using RV2.
- ▶ Send email notifications, such as of error conditions or a batch stopping.

You can run NENU in three ways:

- ▶ Manually using the NENU Manager
- ▶ Automatically using the Windows Task Scheduler, either at scheduled times or when triggered by a system event
- ▶ Automatically as a task of the workflow of an application

Figure 3-34 shows the NENU Manager window.

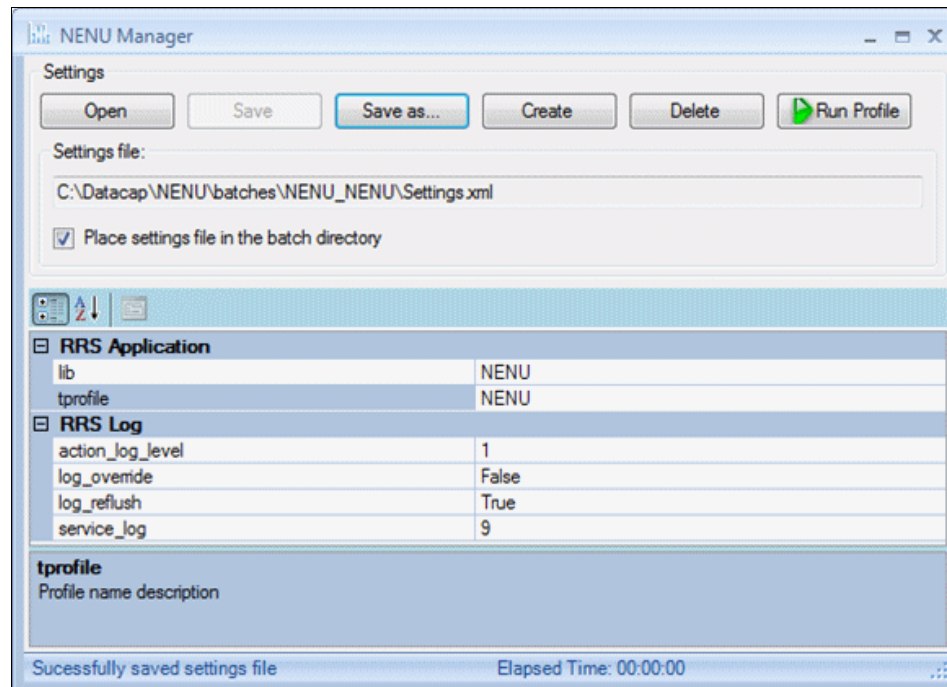


Figure 3-34 NENU Manager window

3.8 FileNet Content Manager for production imaging

IBM FileNet Content Manager provides the *repository* functionality to store, secure, organize, index, and retrieve images and data that are extracted by Taskmaster Datacap. It also provides a *document-centric workflow* infrastructure to automate the routing and processing of these documents.

FileNet Content Manager has a range of advanced capabilities to manage electronic documents and their life cycle. It can handle large volumes, making it a robust platform for document imaging.

As indicated in Chapter 1, “Production imaging overview” on page 3, FileNet Content Manager typically comes into play in the post-committal stage of the document imaging process. You use it when the Taskmaster process completes by storing the captured documents and their metadata in the repository for processing by business users.

At a minimum, you typically use the following FileNet Content Manager functionality:

- ▶ One or several libraries, or *Object Stores*, to host and organize logically the document collections and to provide the document management logic for users to query and retrieve documents
- ▶ *File Storage areas* to provide storage for the images
- ▶ *Document classes* that match the business artifacts that are processed in Taskmaster (such as claims and invoices)

Document classes define the metadata, or *properties*, that are needed for retrievals, but also for processing the documents in the business process. Document classes in that context are also seen as a vehicle for transferring data from Taskmaster to the workflow itself.

- ▶ *Workflows* that define the *work items* parceled out to business users and the processing routes to be followed

Workflow definitions include the data fields, steps, routing conditions, document attachments, and so on. Work items are distributed to the inboxes of users in FileNet Workplace XT. In a single interface, they bring together the data and documents that users need to perform their tasks. All aspects of a workflow definition are specified in a workflow map. A *workflow map* is an XML document that is stored in the repository and used by the Process Engine as a template to instantiate a workflow at run time.

- ▶ *Workflow subscriptions* that define the relationship between a document class and its metadata, a workflow, and a type of event

When an event (such as creation) occurs on the document, a workflow instance is launched with the document automatically added as an attachment. Also the work item fields are populated with the data transferred from the mapped document properties.

For more information about the capabilities of FileNet Content Manager and planning for its implementation, see the following IBM Redbooks publications:

- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547
- ▶ *IBM FileNet P8 Platform and Architecture*, SG24-7667

3.8.1 Workflow management tools

In the course of implementing a Production Imaging Edition project, you must use the following workflow management tools, which are included in FileNet Content Manager:

- ▶ *Process Designer* to design the workflow maps and optionally *Microsoft Visio Connector* to conveniently edit workflows in Microsoft Visio
- ▶ *Process Administrator* and *Tracker* to search and provide the status of executing workflow instances
- ▶ *Case Analyzer* to gather workflow statistics
- ▶ *Process Simulator* to project execution of workflow in production

The following section provides high-level information about what these tools are used for and how they participate in the configuration of a Production Imaging Edition implementation. For more information, see *Introducing IBM FileNet Business Process Manager*, SG24-7509.

Process Designer

Process Designer is used to design the workflows that drive the processing of Production Imaging Edition documents after they are committed to Content Manager. In addition to delivering Production Imaging Edition documents to business users as attachments, the workflow work items carry data that is extracted and verified by Taskmaster. The work items also carry data that constitutes the intelligence needed for applying efficient routing logic and for users to make informed decisions.

The Process Designer defines the activities and resources that are required to accomplish a particular business process. They are represented as a graph, with

a series of process activities, or steps. These activities are connected by routes, or vectors, that define the sequence in which the steps are run and the transition conditions that must be satisfied for routing. Steps and routes are organized into reusable maps. Figure 3-35 shows a claim process map in the Process Designer.

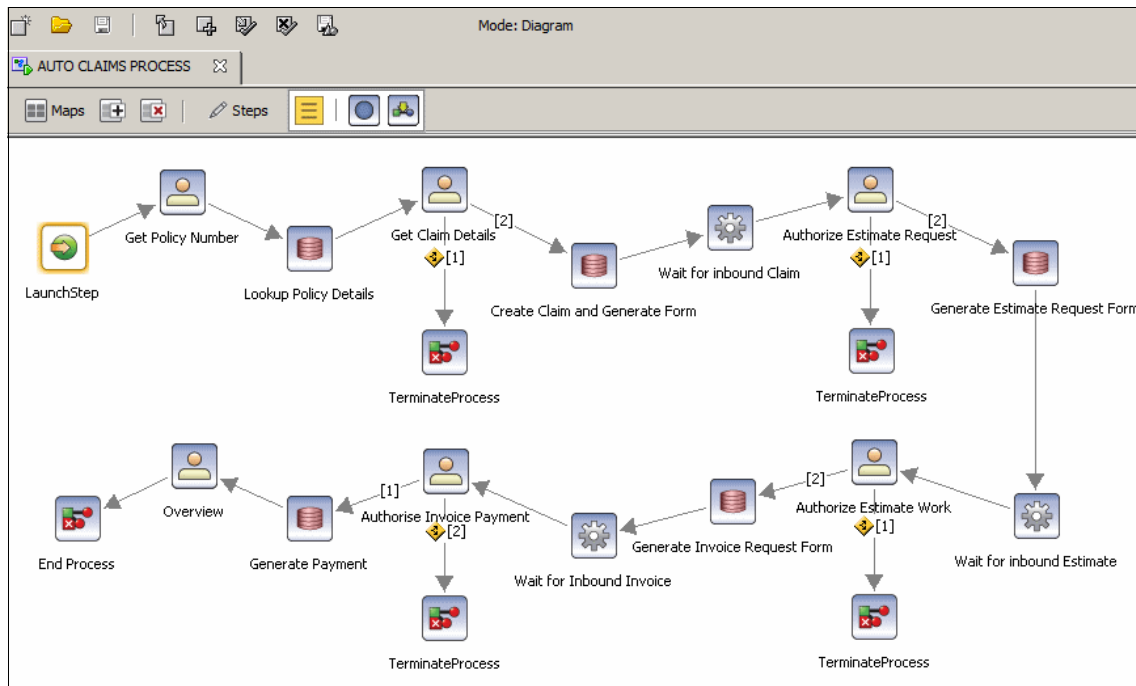


Figure 3-35 Claim process map in Process Designer

Process Designer defines the following items among others:

- ▶ The instructions to complete the task at each step
- ▶ The sequence of steps and step processors (interfaces)
- ▶ The data fields and attachments of the work item and how they are exposed and used by step processors and users at each step
- ▶ The routing and transition logic that is used on arbitrary responses or the data fields to advance the workflow from step to step along the business process
- ▶ The users and roles, system process, or shared work queue that is assigned to each step
- ▶ Timers and the deadlines for completing tasks

Process Designer also provides the capability to validate and run a workflow in real time. It and saves the process definition in the repository in the form of an

XML Process Definition Language (XPDL) file. It can also import existing processes from IBM WebSphere® Business Modeler or Microsoft Visio.

Microsoft Visio Connector

With the Microsoft Visio Connector, you can use Microsoft Visio to create diagrams of business processes. Then you can import them into Process Designer as a starting point for creating a workflow.

For business analysts who are familiar with Business Process Modeling Notations (BPMN), a Visio stencil of BPMN shapes is provided with each shape. The stencil is mapped to a corresponding Process Designer object, such as a step, route, text annotation, data object, Workflow Group, or submap. Shapes in standard stencils are also mapped to Process Designer objects. You can customize the mapping for shapes on existing Visio diagrams.

On the left side of Figure 3-36, you see the standard shapes that can be used to create a standard Visio diagram and how it looks in Process Designer after it is imported. On the right side, you see the BPMN notations stencil and a sample Visio workflow with submaps and how they look in Process Designer.

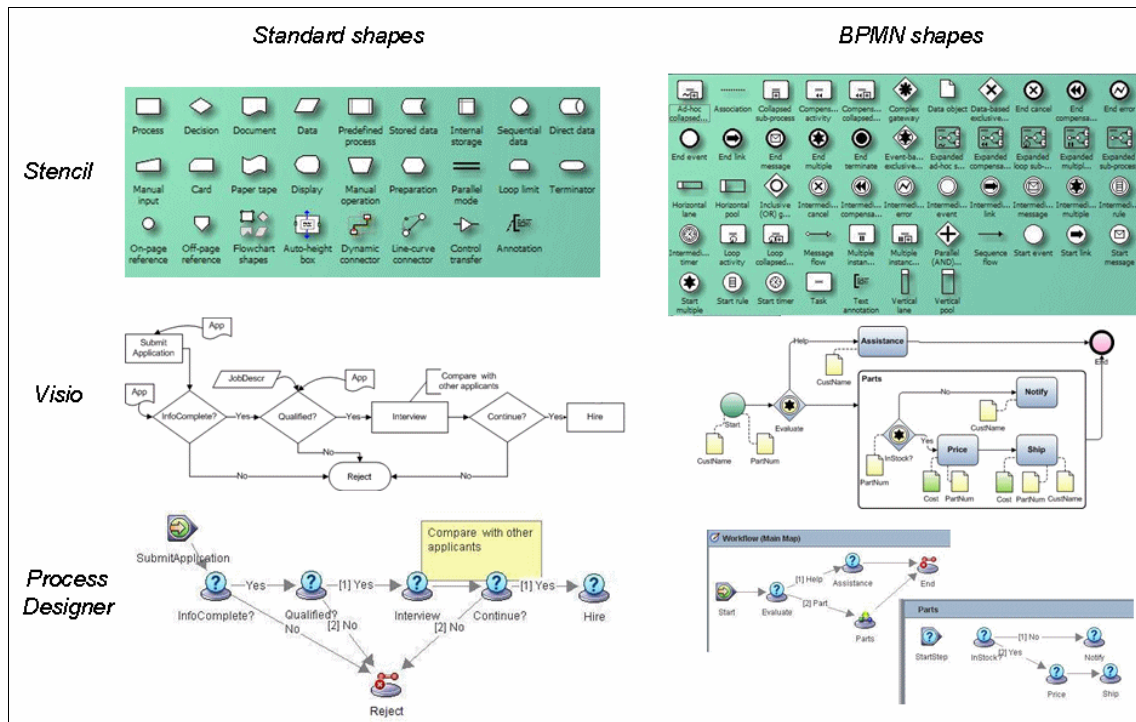


Figure 3-36 Process maps and Microsoft Visio connector

Process Administrator and Process Tracker

Process Administrator (Figure 3-37) is used to manage executing workflows, search and view specific instances, and edit their data and properties while they are running. Process Administrator is a powerful tool that workflow administrators use to monitor work items and resolve processing issues.

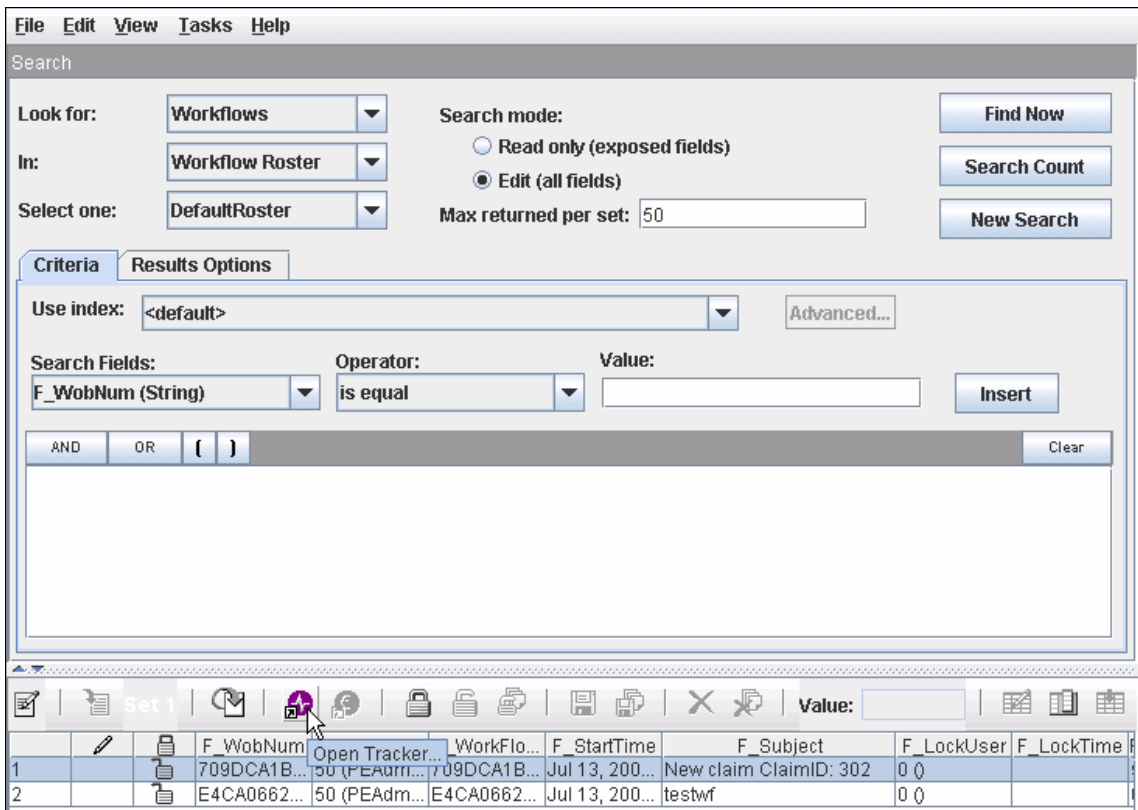


Figure 3-37 Process Administrator

After you select a specific workflow instance, you can view its status in the Process Tracker. Process Tracker provides the status of a currently running workflow in a graphical view by using the workflow map of the process definition created in the Process Designer (Figure 3-38).

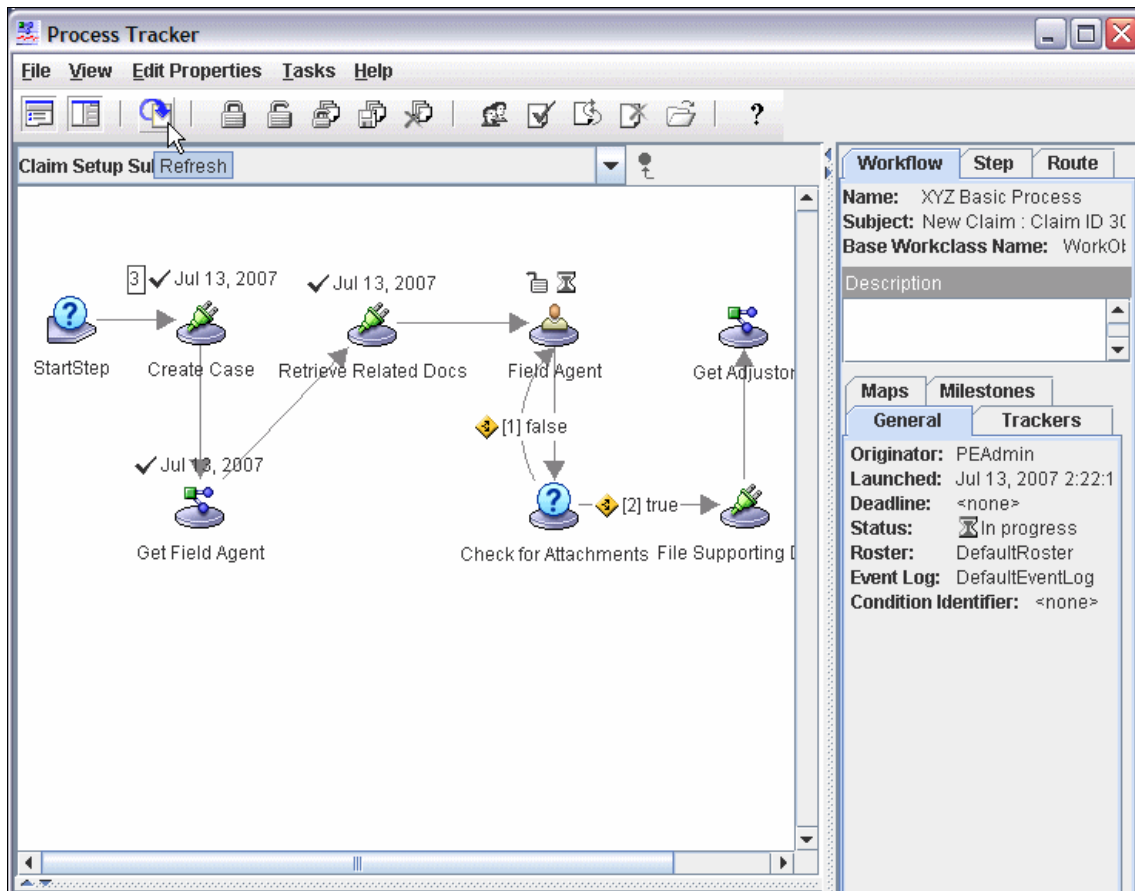


Figure 3-38 Process Tracker displaying the runtime status of a workflow instance

From the graphical view, you can tell which steps have been completed in the workflow, when they were completed, and which steps are currently active.

Case Analyzer

Case Analyzer is used to discover business processing trends (as opposed to looking at specific instances to establish a baseline for measuring productivity) and to identify bottlenecks. It monitors and analyzes large numbers of events generated by the Process Engine, which it aggregates for analysis in an OLAP cube stored in a Microsoft SQL server database at regular intervals.

For reporting, Case Analyzer uses Microsoft Excel by default, which has a rich set of chart and reporting functions for viewing and analyzing the data stored inside the OLAP database.

The Case Analyzer comes with the following set of standard reports:

- ▶ Productivity, which measures wait, processing, and completion times of the work items in the various steps and queues (Figure 3-39)
- ▶ Queue Load, which measures the number of work items added, completed, and currently left in a particular queue or step
- ▶ Workload, which measures the number of workflow runtime instances created, completed, and currently executing, and which measures the average processing time of the various workflow instances
- ▶ Work in progress, which provides a real-time view of the currently active work items by measuring their count in various steps and queues and by measuring the time that the work items have spent at the current step
- ▶ Workflows in progress, which provides a real-time view of the currently active workflow instances by measuring their count and duration

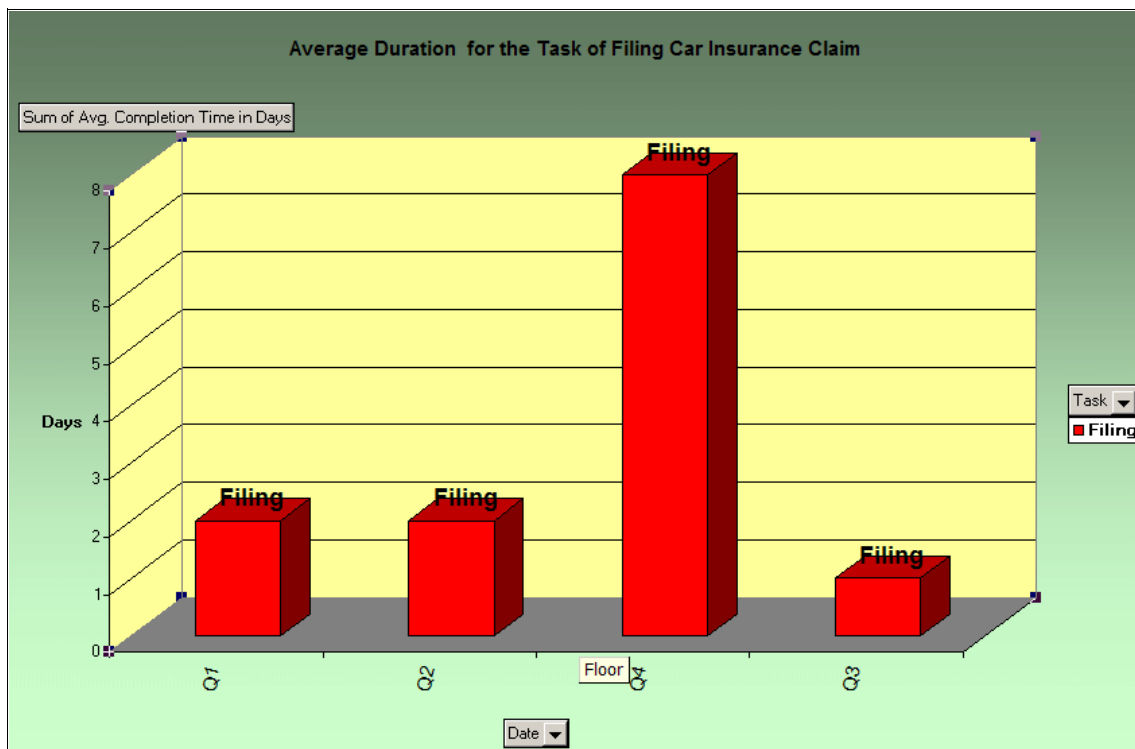


Figure 3-39 Sample Case Analyzer report

You can also create your own custom reports by using IBM Cognos® Analysis Studio or IBM Cognos Reports if you require more sophisticated reporting capabilities.

Process Simulator

You use Process Simulator to validate a process before placing it into production. Process Simulator can be used in two scenarios. The first scenario consists of validating a new process, where the user provides the information for the simulation based on the experience and knowledge of the user about the domain. The second scenario consists in validating enhancements to an existing process, where historical data from Case Analyzer is used to feed the simulation.

In either case, the user can simulate what-if scenarios and then analyze the results of the simulation to validate the process. That way, a process analyst can test different scenarios to improve the business process before deploying it in production.

You access Process Simulator from the Process Simulation Designer and the Process Simulation Console. The Designer is used to create simulation scenarios that can be saved in the Content Engine repository. Process Simulator supports versioning for the saved scenarios.

The Process Simulator Console is used to manage the scenarios and to execute simulations that have been created. The results of a simulation execution are stored in the Case Analyzer database. The simulation object itself, which contains animation information, is stored in the Content Engine for future animations. Animations are run in the Animator as shown in Figure 3-40 on page 113.

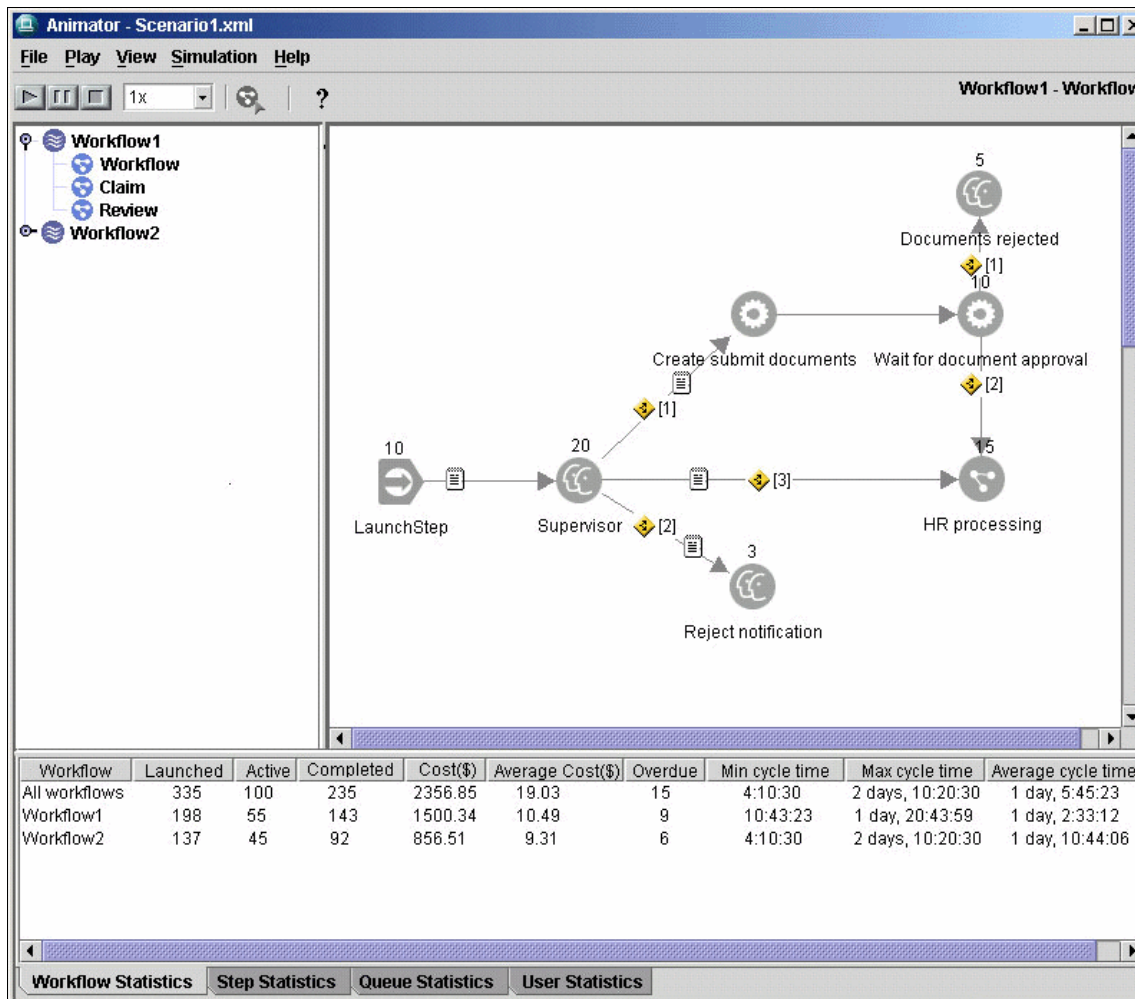


Figure 3-40 Process Animator running a simulation

3.9 Advanced production imaging viewing

The IBM Production Imaging Edition Viewer is based on Daeja ViewONE Pro. It expands on the functionality delivered in the standard viewer of FileNet Content Manager with the following features:

- ▶ PDF viewing and annotating directly in the viewer
- ▶ Universal viewing and annotating of electronic documents

- ▶ Document streaming to improve response time when displaying large documents
- ▶ Permanent redaction to burn redactions and annotation to TIFF and PDF

3.9.1 PDF viewing and annotating

With the PDF module of the Production Imaging Edition Viewer, you can view PDF documents without having Adobe Acrobat Reader installed.

You can view Acrobat annotations that have been saved to the PDF document, but you cannot edit them. However, you can add Daeja annotations to the PDF documents that are stored outside the document, similar to other annotations on images in FileNet Content Manager (Figure 3-41).

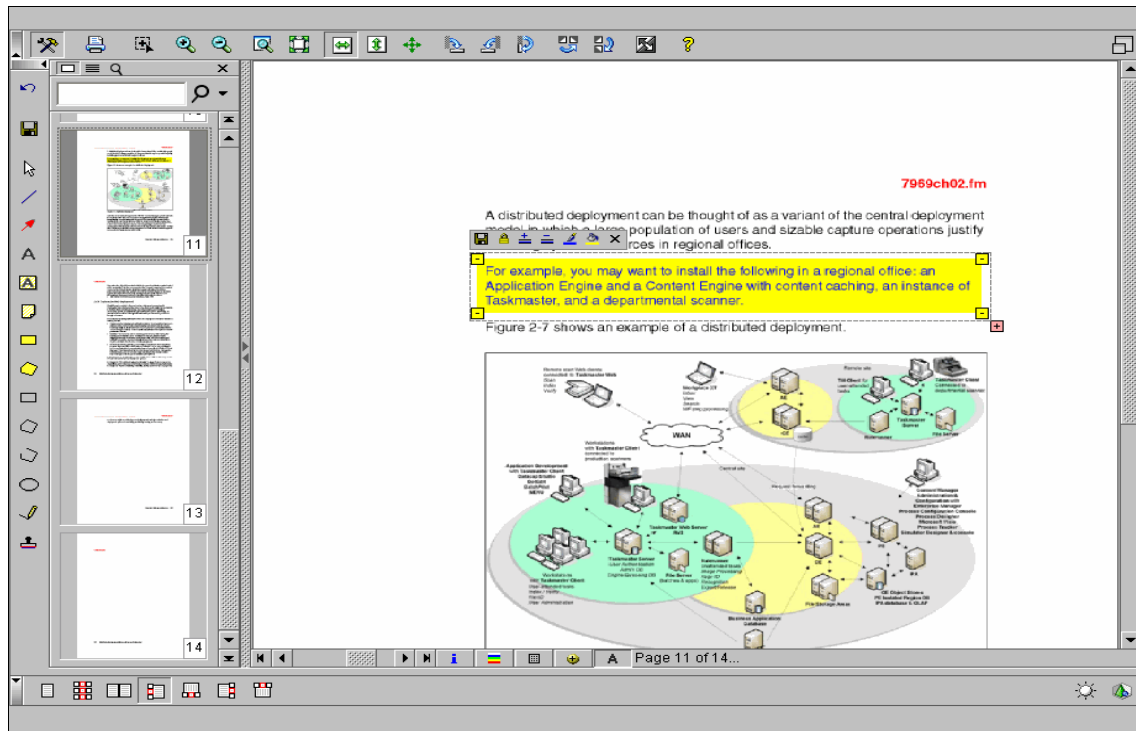


Figure 3-41 Adding an annotation in a PDF document

You can also search the contents of PDF documents that have text in them by using a search box. Daeja finds all the occurrences of the word you are looking for and highlights them in the page that you display.

The streaming module works with searches. This way, when you search a large PDF document, the search is deferred to the streaming server with only the results. Also the page numbers embedded within them returned to the client, so that the viewer knows which page to jump to. When selecting a given page, the viewer working with the streamer component only retrieves that page. It does not download the entire document on the client, which happens otherwise with a standard PDF document download.

Redaction is also possible on PDF documents. Permanent redaction creates a PDF document with the redacted area secured in such a way that the text is removed and is not searchable in the output PDF.

3.9.2 Universal viewing and annotating

The Universal Viewing module provides the Production Imaging Edition Viewer with the capability to view and annotate Office documents and many other electronic document formats.

Similar to PDF, you do not need to have and launch the native applications every time you want to display a document. You can browse through TIFF, PDF, and Microsoft Office documents in work items without switching applications. You can annotate them in the same way as you do with TIFF documents.

No installation is required on the client for the Universal Viewing module to work. It is downloaded transparently to the local cache folder the first time you view a supported electronic document.

3.9.3 Document streaming

The streaming module is used to reduce network traffic and improve the performance perceived by the users. The largest impact of this feature is on TIFF and PDF documents that have a higher number of pages. It splits up the document into individual pages and sends each page to the client on demand.

The streaming module has a servlet in FileNet Workplace XT (see the architectural details in Chapter 2, “System architecture” on page 35) that caches the full document, but only delivers the pages requested by the Production Imaging Edition Viewer. When used with PDF, it also compresses the PDF pages that are sent to the Production Imaging Edition Viewer applet.

Also, as indicated previously, streaming also works with searches so that the search is run by the streaming servlet. In turn, the servlet sends, to the Production Imaging Edition Viewer applet, the results with the page index for the viewer to know which page to get from the streaming cache.

3.9.4 Permanent redaction

The Production Imaging Edition Viewer also provides a permanent redaction capability. The primary usage is to redact the areas of a document that you do not want to make public, such as a social security number. The Viewer permanently “burns” the redactions in the document so that the information cannot be recovered and you can safely distribute a copy of the document.

Redaction is supported on any format that can be viewed. For TIFF and electronic documents, such as Microsoft Office documents, the redacted document is output to TIFF (see the example in Figure 3-42). For PDF, the redacted document is output to PDF.

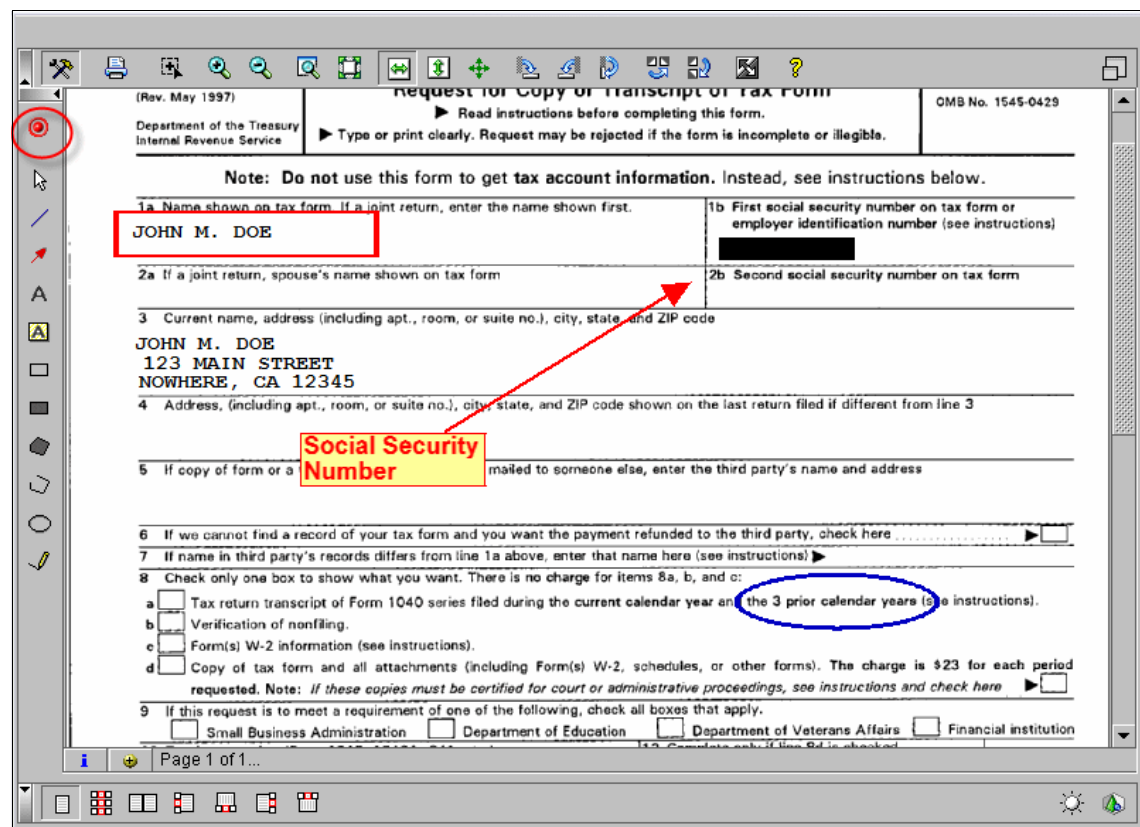


Figure 3-42 Redacting a TIFF document in the Production Imaging Edition Viewer

The source PDF document might include text, such as with electronic documents converted to PDF or image documents that have been recognized by using OCR results saved to PDF. In this case, the text of the redacted areas is securely

removed by the redaction and cannot be searched or recovered. However, the rest of the PDF content is not affected and can be viewed, and the text is used in searches.

When used with TIFF and electronic documents, the burning part of the redaction process works with most annotations, so that you can create a permanent annotated copy of a document.

With PDF, you can only use rectangular fill areas to redact the document, and the other annotations are not available for burning.

The redaction process is initiated from the **Redact to File** menu on the document selected in FileNet Workplace XT and sends the redacted output file to the client to be saved locally.

Similar to the Streamer component, the Permanent Redaction component is implemented as a servlet. It is invoked from the Annotation module in the Production Imaging Edition Viewer applet to do the burning to TIFF or PDF and send the output file to the client. (For architectural details, see Chapter 2, “System architecture” on page 35.)

Production Imaging Edition Viewer redaction versus Datacap

Taskmaster redaction: Production Imaging Edition Viewer redaction is different from the redaction capability offered in Datacap Taskmaster. Taskmaster redaction is typically applied at the precommittal stage where the original, unredacted document, might or might not be stored in the repository. The Production Imaging Edition Viewer redaction is applied manually on documents at the postcommittal stage. The resulting redacted content is saved on the client without affecting the source document in the repository.

3.10 Bulk Import Tool

The Bulk Import Tool is a new utility that ships with FileNet Content Manager 5.1 and later. It provides the capability to ingest large volumes of documents quickly in one or several Content Engine Object Stores.

Licensing: Customers can use the Bulk Import Tool as part of the license for Production Imaging Edition.

For example, you can use it in a production imaging scenario for the following purposes:

- ▶ Decoupling scanning operations from the back-end ECM system, as is the case when subcontracting digitization to a service bureau
- ▶ Migrating large volumes of documents from an existing system
- ▶ Ingesting automatically machine-generated documents, such as utility statements

The Bulk Import Tool monitors file system directories for batches of documents to import. It looks specifically for the batch description files (<batchnumber>.eob) that it uses to create corresponding batch import entries in the Content Engine import queue. Then it triggers the import process. Figure 3-43 shows the Bulk Import Tool processing model.

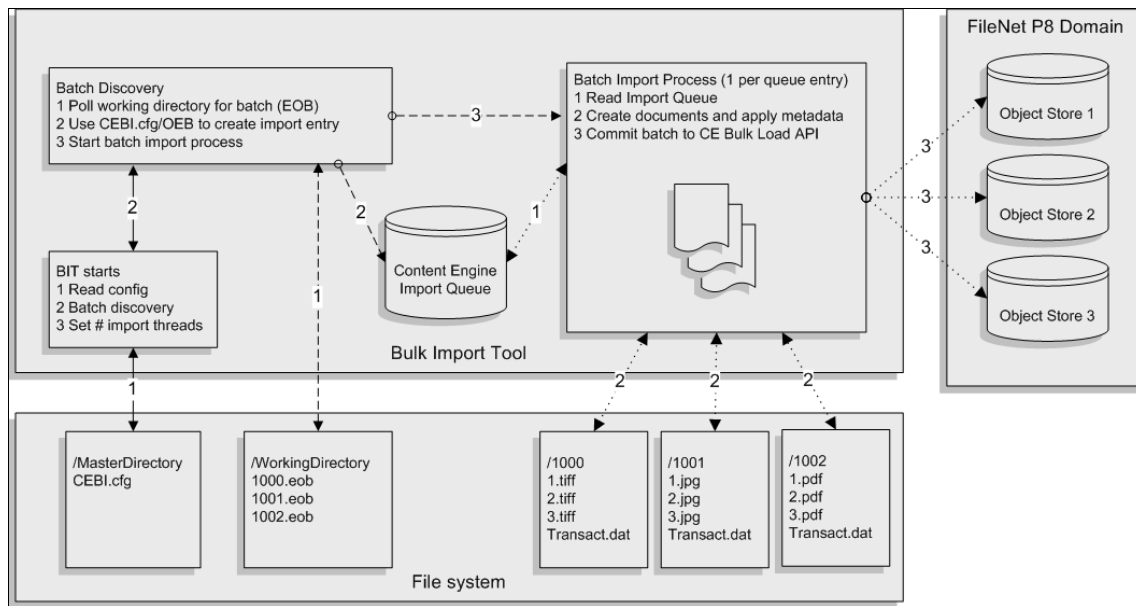


Figure 3-43 Bulk Import Tool processing model

The batch import process reads the queue entry information of Content Engine that points to the location of the metadata description file (transact.dat). This file accompanies each batch and contains the index information of each document with the references to the content files (for example, TIFF files). It then creates and indexes all the documents in the batch and commits it to the Content Engine. For each batch, the Bulk Import Tool generates a journal file that logs the results.

You can configure the Bulk Import Tool to run multiple instances, each configured for a specific Object Store, in addition to batch size, sleep time between runs, and number of background threads to Content Engine. This flexibility provides increased throughput and helps you to tune the consumption of system resources.

The Bulk Import Tool has been modeled after the High Performance Image Import (HPII) utility, which has been used by FileNet Image Services customers for years. It offers near compatibility with earlier versions of HPII so that customers who currently use HPII for Image Services can use the Bulk Import Tool with minimal changes to their current environment.

3.11 Conclusion

This chapter focused on the key functions, component and tools of Taskmaster, how the process works, and how to build an application. It also highlighted FileNet Content Manager and the advanced viewer.

With this knowledge, in Part 2, “Solution implementation” on page 121, we guide you through an actual implementation of a Product Imaging Edition solution with a use-case solution sample. We begin with the solution design and take you through the step-by-step solution implementation.



Part 2

Solution implementation

This part explains how to design a document image capture solution and how to implement the solution with step-by-step procedures. It introduces a use-case scenario that demonstrates concrete implementation steps.

This part includes the following chapters:

- ▶ Chapter 4, “Solution example” on page 123
- ▶ Chapter 5, “Designing a production imaging system” on page 139
- ▶ Chapter 6, “Implementing the capture solution” on page 179
- ▶ Chapter 7, “Adding a document type to an existing application” on page 253
- ▶ Chapter 8, “Implementing the business process component” on page 269
- ▶ Chapter 9, “Best practices and recommendations” on page 289



Solution example

This chapter describes a scenario where IBM Production Imaging Edition is used to improve an existing business process. This scenario illustrates how the capabilities are applied in a realistic although simplified use case.

This chapter includes the following topics:

- ▶ Scenario background
- ▶ Current claim approval process
- ▶ New claim approval process
- ▶ Summary of benefits

4.1 Scenario background

The fictitious company, Fictional Insurance Company A is a global insurance company, headquartered in the United States with satellite offices worldwide. Company A is looking to improve business processing and to streamline inefficiencies in its claim approval process. The current process is partially manual and time consuming. Although the company has strived to replace paper processes with automated electronic processes, portions of the process are still constrained by physical paper.

More information about claims processing: For information about claims processing, see the IBM Redbooks Publication, *Introducing IBM FileNet Business Process Manager*, SG24-7509. See also the IBM Case Manager solution template, “Auto Claims Management sample solution template for IBM Case Manager,” in IBM developerWorks® at:

<http://www.ibm.com/developerworks/data/tutorials/dm-1101casemanagertemplates2/index.html?ca=drs->

This scenario addresses aspects of the claims approval process. Specifically, it addresses how to streamline the submissions of documentation to the process by using document imaging including capture.

In Company A, the claims process typically starts with a client who is involved in an accident and calls the insurance company. A client service representative takes the call and works with the client to complete a form. In addition to details about the client and the accident, by using the form, the client service representative can specify additional steps. For example, the representative can determine whether a rental car is required or whether a third party might be responsible for part or all of the damage. In the current process, the client service center needs to send a form to the client for the client to sign and enter a date, acknowledging the claim. The client must then return the requested documentation.

Company A seeks to improve their business process by streamlining inefficiencies in their claim approval process. The company is also concerned about increasing incidents of fraud and wants to find new ways to use the information that is currently locked inside of the documents.

The current process depends on the documentation that comes from the field offices or independent agents. Some claims are submitted to field offices or independent agents who forward the paper to the claims center. Some claims are mailed or faxed directly to the claims center. Other claims find their way to the general address or fax number or the company. It can take long time to direct the hardcopy to the right department in the process and to the individual who is responsible for handling the

claim. If the claim has a mistake, such as missing signatures or a missing document, the mistake impacts the entire claims approval process. In addition, the paper claim is at risk for becoming lost in transit. Missing or incomplete documentation can delay the claim settlement resolution for the policy holder.

Other issues have arisen. For example, it is difficult to find and group all of the documents that are associated with a policy holder and their open or closed claims. With a long and error-prone claim approval process, Company A faces unsatisfied clients, the potential of losing clients, and the risk of losing their market share.

Without a consolidated view of the activity of the client, the company is vulnerable to fraud. As fraud becomes an issue, the company wants to make the claim activity more transparent and get early indicators of potential fraud.

The top priority of Company A is to shorten the processing time for claims to improve client satisfaction. Documents need to reach the desk of the claim handler faster. Knowledge workers should be relieved from repetitious, clerical tasks. The process relies on printed documents and requires people in different locations to make decisions based on the content. Production Imaging Edition is the perfect solution to make the content available and to streamline the dependent business processes.

4.2 Current claim approval process

This section describes an extremely manual process. It outlines a worst-case scenario for the current process, illustrating all of the areas where improvement might lie. In your environment, you might have applied some improvements already in your process. Therefore, you might find this description less efficient than your current process.

Imagine that a client has a car accident. The car was severely damaged and towed to a repair shop. Fortunately, no one was injured. The client calls the local agent to report the accident and to make a claim. The client informs the agent that the car was towed to the repair shop.

The agent sends a standard form to the client to gather information about the incident. The client, named Jane, completes the form with information about herself, the policy, and the accident. She signs the form, enters the date on the form, and then sends the completed form by mail or fax to the agent. The agent checks the form for completeness and manually enters the claim information into the claims system. Then, the agent places the paper in an envelope and mails it to the Company A Claims Center.

The claim form is received in the mailroom of claims center, which receives tens of thousands of documents daily. The mail is logged, opened, date-stamped, sorted, counted, grouped, and placed in carts. The carts are rolled to the claims floor. The claim documents go to the filing clerk. The filing clerk takes each claim document, records it in the claim system, prints a folder label and attaches it to a file folder, and places the claim documents in the folder. The folders are put in a cart and distributed to inboxes on the desks of the claims handlers.

A claims handler takes a claim folder from the inbox, looks up the claim on the workstation, and enters additional claim information from the form into the claims system. If a problem arises, such as a missing signature, the claims handler completes an exception routing slip, attaches it to the folder, and places the folder in an exception out box. The good claims are placed in a separate out box for filing.

Periodically a filing clerk takes the folders from the out boxes. The good claims are taken to a filing clerk who files the document in a suspense folder in the claims center. The exceptions are taken to the exception desk for handling.

If a signature is missing, the clerk at the exception desk makes a copy of the claim form and places it in a window envelope and places the envelope in an out box for outgoing mail. The claims folders are placed in an out box for file folders that need to be filed.

Periodically a filing clerk takes the files from the out boxes. The claims folders are taken to a filing clerk who files them in a suspense file in the claims center. The mail is taken to the mailroom where, twice a day, the outgoing mail is sent through the postal system.

Meanwhile at a remote office, an agent checks if the repair shop, where the car has been towed, is authorized to perform repairs for Company A. Fortunately, this repair shop is authorized to perform repairs. The agent requests a written estimate from the repair shop.

The repair shop sends an estimate back to the agent by mail or fax. The local office mail clerk delivers mail and faxes to the desk of the agent. The agent goes through the mail several times a day and reviews and approves estimates. The agent checks the status of the claim online to see if the claims center has received the completed claim form. If the system shows that the form was received, the agent approves the repair on the system. The agent notifies the repair shop that the repair has been authorized and makes a copy of the estimate to keep on file. Once a day, the agent sends the original estimate documents to the claims center to be filed with the claim.

The claims system has a letter generation job that also sends a letter to the client, indicating that the claim is approved, and to the repair shop, indicating that

it is authorized to begin repairs. Several days have passed since the client reported the incident.

The estimate from the repair shop is received in the claims center mailroom, sorted out, and placed in a cart that is rolled to the claims floor. The estimates go to the filing clerk who pulls out the claim file folder and adds the estimate to the folder. If the folder is missing (for example, pulled by a client service agent), a second folder is created to hold the estimate until the claim folder is returned to the file clerk. The estimate sits in the claim folder with the claim documents until the repair is completed.

After the car is repaired, the repair shop sends an invoice to Company A for payment. Two days pass before the claim center mailroom receives the invoice. The invoice document is received in the mailroom, sorted out, and placed in a cart that is rolled to the claims floor. The invoice goes to the filing clerk who pulls the claim file folder and adds the invoice to the folder. The filing clerk puts folders with invoices and estimates on a cart and delivers them to inboxes on the desks of the claims handlers. If there is no matching estimate, the file is taken to the exception desk with an exception routing slip.

The claims handler looks up the claim on the workstation and enters the invoice number, invoice date, and invoice total into the claims application. The claims system determines if the estimate and invoice match and if payment is authorized. If payment is not authorized, the clerk completes an exception routing slip, attaches it to the folder, and places it in an exception out box. The folders for good claims are placed in a separate out box for filing.

Periodically a filing clerk takes the completed documents from the out boxes. The good claims are returned to the suspense folder in the claims center. The exceptions are taken to the exception desk for handling.

Once a month, the completed claims are placed into boxes by claim number. The boxes are transported to the records center for long-term retention.

It is apparent that this process is long and involves many people who handle, transfer, match, and file the documents; who enter data; and so on. The goal with Production Imaging Edition is to eliminate as many of these tasks as possible so that the process is simpler and faster.

4.3 New claim approval process

To improve the claim approval process, all the elements of a production imaging system are applied. This process captures documents and digitizes the paper form. Alternatively, the process can avoid paper altogether and use electronic forms when a physical copy is not needed.

Some documents still come from external parties or require signatures on the paper form. Therefore, Company A continues to use paper in these instances. The company receives the documents by scanning, fax, or email. It uses Optical Character Recognition (OCR), Intelligent Character Recognition (ICR), and Optical Mark Recognition (OMR) recognition to read data directly from the documents to eliminate manual typing of the data. The company stores the documents in the imaging system so that they are available online to anyone who is involved in the claims process.

The company uses workflow and process automation to streamline the process and eliminate manual sorting, routing, and filing steps.

Figure 4-1 illustrates the new claim approval process.

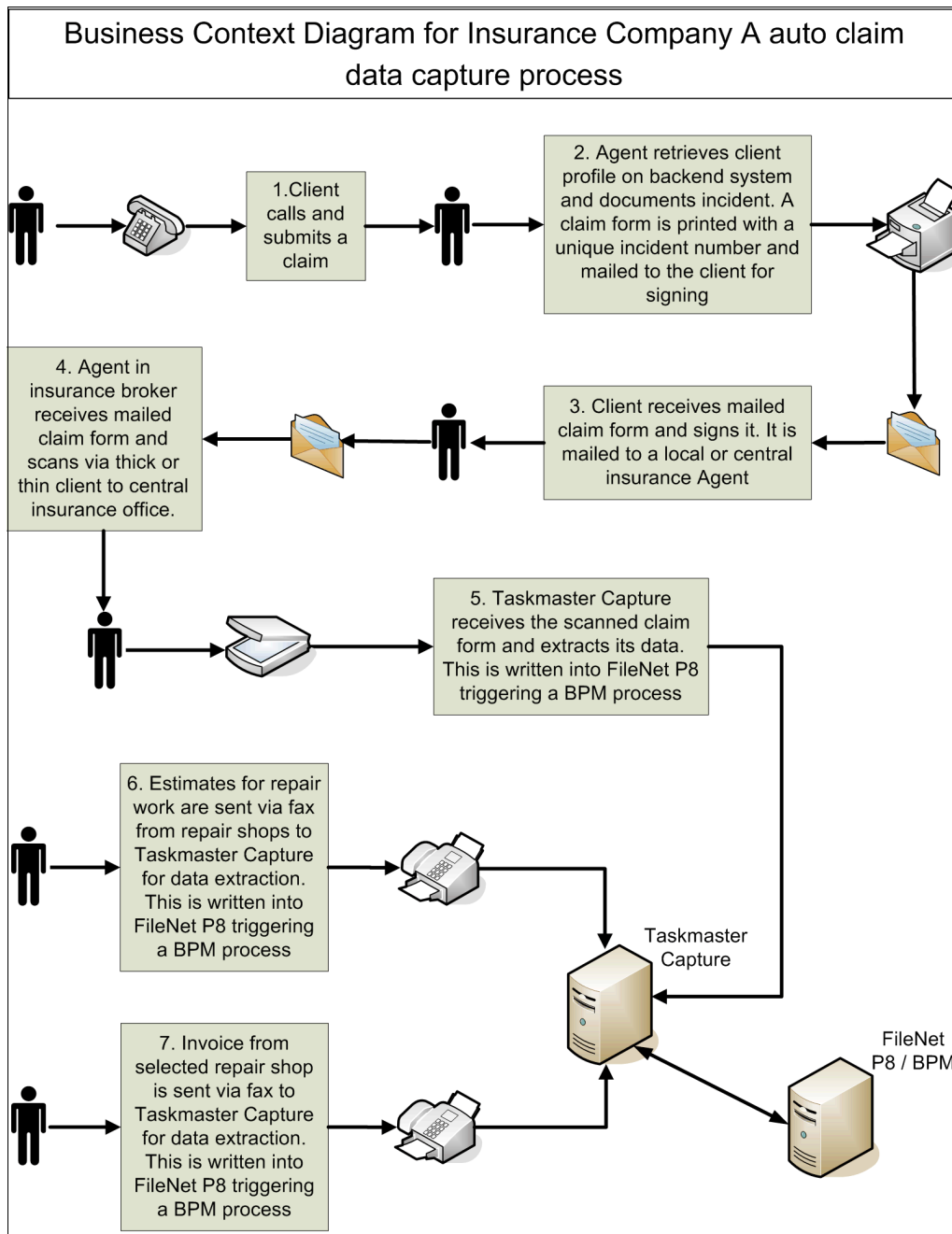


Figure 4-1 Business context diagram of the automatic claim process

The new system can react to changing business requirements or modified infrastructure constraints quickly and efficiently. The architecture reuses existing components by orchestration.

This solution takes advantage of the following capabilities:

- ▶ Receiving documents that are scanned or sent by email or fax that are distributed from a central location
- ▶ Automatically identifying and classifying documents
- ▶ Lifting data from documents with OCR, ICR, OMR, and barcodes
- ▶ Validating data using external data sources
- ▶ Performing logical validation
- ▶ Checking for signatures
- ▶ Cataloging and storing documents in an imaging repository
- ▶ Triggering business processes using active content
- ▶ Automating content-centric business processes

In the new process, the client can either call an agent or enter the incident online through the claims portal. In either case, electronic form records the client information.

A claim form is printed with barcodes to identify the form type and the policy data of the client. The client does not need to enter this data manually. The system mails the form to the client or sends it by email with a link to download and print the form. If clients file the form online, they can print the forms by themselves and sign them. Then they can return the signed form by fax or by email.

Figure 4-2 shows a sample of a printed claim form.


Auto Insurance Claim Form Insurance Company A	
 <small>*CLAIM-20A*</small>	
Policy Holder Address :	John Doe, 123 Springfield Drive NY 234567
Driver Name :	John Doe
Policy Number :	43398313
Incident Number :	CL-31
Incident Date:	03/02/11
Incident Time:	13:30
Vehicle License :	5566TY
Vehicle Colour :	Silver
Vehicle Manufacturer :	Lexa
Vehicle Model :	Charger
Year of Manufacture :	2008
Chasis Number (VIN) :	2B3AA4CTX82KP3456
Incident Description:	
Incident occurred on Highway 42, NY. Damage occurred to front bumper and front right headlight. Airbag was triggered.	
Was anyone injured in this incident which required medical attention?	<input type="checkbox"/> Yes <input type="checkbox"/> No
<div style="display: flex; justify-content: space-between;"><div style="border: 1px solid black; padding: 5px; width: 45%;">By signing the adjacent box and entering the current date, you agree the information above is accurate to the best of your knowledge.</div><div style="width: 50%; text-align: right;"><div>Signature: <div style="border: 1px solid black; height: 40px; width: 100%;"></div></div><div>M M D D Y Y Dated: <div style="border: 1px dashed black; display: inline-block; width: 100px; height: 20px;"></div></div></div></div>	

Figure 4-2 Printed claim form

Meanwhile, the business process is initiated that handles the rest of the steps in the process. Alternatively, a case can be initiated with IBM Case Manager.

Most of the information on the claim form is prefilled. In many instances, the client only needs to sign the form and enter a date to acknowledge that it is correct. The form contains a check box that the client manually selects if there is a medical injury. The client can send a scanned copy of the form by mail, email, or fax to the claims center or to the agent. Because some clients prefer personal service from an agent, the client has the option of working directly with their agent.

If the form goes to the agent, the agent can use web-based remote scanning to scan the document to the claims center. Documents that are mailed to the claims center are centrally scanned. Documents that are faxed are received by a fax server, and a connection to the fax server loads the fax into the capture system. Documents that are sent by email are received in a service inbox, and a connection to the email system loads the email message into the capture system.

Regardless of the input channel, the process is the same. A capture workflow job is initiated to process the document. The capture system identifies the type of document and the specific incident by reading a barcode. The handwritten date is read using ICR. The system examines the area where the client signs the form. If it is blank, it is flagged for manual handling.

The claim form and any other attached documents are electronically filed in the imaging repository. Each document is identified with the incident number, policy number, and other key data that makes it easy to search and retrieve the documents. With the current solution, the document is available for viewing by every stakeholder in Company A without pulling a paper file or getting a copy.

The receipt of the form performs a second critical role. It triggers the claims process that was waiting for the document. The act of storing the document notifies any waiting processes that the document has arrived. This content is *active*. That is, the document is aware of related business processes and interacts directly with these processes. As a result, the claims process continues automatically when the document is received.

If an exception occurs, an automatic subprocess is initiated. For example, if the claim form was not signed by the client, a subprocess generates an email message or letter and sends it to the client asking the client to sign and resubmit the form.

Process automation can be applied to orchestrate the interaction with the agent, adjuster, and repair shop. For more information about this type of automation, explore the following resources for related scenarios:

- ▶ *ACI Worldwide's BASE24-eps V6.2: A Supplement to SG24-7268*, REDP-4338
- ▶ IBM Case Manager solution template, "Auto Claims Management sample solution template for IBM Case Manager" which is available on IBM developerWorks at:
<http://www.ibm.com/developerworks/data/tutorials/dm-1101casemanagertemplates2/index.html?ca=drs->

Meanwhile, at a remote office, an agent checks whether the repair shop, where the car was towed, is authorized to perform repairs for Company A. In this scenario, this repair shop is authorized to perform repairs at contracted rates. The agent requests a written estimate from the repair shop.

The repair shop faxes an estimate to the fax server of Company A. The capture software imports any documents received by the fax server. The system recognizes that the document is an estimate, extracts key information, looks up the claim number, and matches the estimate to the claim.

Occasionally an unknown form is received. For example, a new repair shop might send an estimate in a new format for the first time. The document is manually identified by using the Click'n Key capability by a verification operator. After the form is identified, it is automatically handled the next time the company receives a similar estimate from this repair shop.

Figure 4-3 shows a sample of the estimate document.

Company B

1234 Field Road, NY 2445

Estimation for Repair Work

Quote for :

John Doe,
123 Springfield Drive
NY 234567

Dated :

11/02/11

Vehicle License :

5566TY

Incident No :

CL-2389534

VIN No :

2B3AA4CTX82KP3456

Our Ref :

VBDA-12333

Manufacturer :

Lexa

Model :

Charger

Please see the below itemised estimation for work on the vehicle detailed above.

Description	Unit	Unit Cost	Cost
Replacement Right Headlight – CH44	1	120.45	120.45
Replacement Bumper 223	1	100.00	100.00
Airbag Replacement	1	100.00	100.00
Respray	1	30.00	30.00
Labor	2	45.00	90.00

Sub :

\$440.45

Tax :

\$35.24

Total :

\$475.69

Figure 4-3 Estimate document

Figure 4-4 shows two potential estimates documents, one of which has multiple pages.

Company V

345 Kilano Grange, NY 3345

Estimate for Repair Work

Quote for :
John Doe,
123 Springfield Drive
NY 234567

License :
5566TY

Dated :
10/02/11

VIN No :
2B3AA4CTX82KP3456

Incident No :
CL-2389534

Manufacturer :
Lexa

Our Ref :
BDAV-12333

Model :
Charger

Please see the below itemised estimation for work on the vehicle detailed above.
This is for estimate purposes only.

Description	Unit	Unit Cost	Cost
Right Headlight fixing kit	1	15.45	15.45
Right Headlight bracket	1	25.00	25.00
Headlight screw pack	1	9.00	9.00
Right Headlight glass panel	1	35.00	35.00
Headlight wire pack	1	8.00	8.00
Headlight connection pack	1	9.00	9.00
Bumper fixing bracket	1	24.00	24.00
Bumper screw pack	2	4.00	8.00
Bumper 3445	1	45.00	45.00
Airbag AI34	1	65.00	65.00
Airbag Cylinder	1	20.00	20.00
Right Front wheel arch 897774	1	75.00	75.00
Respray Paint Color 2EU	1	20.00	20.00
Labor – Per Hour – Reset of Airbag	1	20.00	20.00

Continued Over Page

Continued

Description	Unit	Unit Cost	Cost
Per Hour – Fixing of headlight	1	20.00	20.00
Per Hour – Fixing of Bumper	2	20.00	40.00
Per Hour – Fixing of wheel arch	3	20.00	60.00
Per Hour – Respray	1	20.00	20.00

Sub :

\$518.45

Tax :

\$41.48

Total :

\$559.93

Quote for :
John Doe,
123 Springfield Drive
NY 234567

Dated :
10/02/11

Incident No :
CL-2389534

Our Ref :
BDAV-12333

License :
5566TY

VIN No :
2B3AA4CTX82KP3456

Manufacturer :
Lexa

Model :
Charger

Please see the below itemised estimation for work on the vehicle detailed above.

Description	Unit	Unit Cost	Cost
Right Headlight – CH44	1	120.45	120.45
Airbag Replacement Kit	1	130.00	130.00
Body Front Bumper 223	1	100.00	100.00
Respray Work	1	30.00	30.00
Labor Work	3	45.00	135.00

Sub :

\$515.45

Tax :

\$64.43

Total :

\$579.88

Figure 4-4 Estimate documents, one of which has two pages

Chapter 4. Solution example 135

The system can extract detailed line items from the estimate. In the past, this was too time-consuming and labor intensive. Now the system can do it electronically, and the additional data can be sent to a fraud defense process for statistical evaluation using analytics.

The estimate document is electronically filed in the imaging system with the claim form. The estimate data is updated in the claims system electronically.

The receipt of the estimate document also triggers the workflow to notify the agent. Through an online work portal or mobile application, the agent sees a notification in an electronic inbox. The agent approves the repair and notifies the repair shop that the repair has been authorized.

The claims system letter generation job is still used to send a letter to the client, indicating that the claim is approved, and to the repair shop, indicating that it is authorized to begin repairs.

After the car is repaired, the repair shop sends the invoice by fax to the central fax server. The capture software recognizes that the document is an invoice, extracts key information, looks up the claim number, and matches the claim to the estimate.

The invoice document is electronically filed in the imaging system with the estimate and claim form. The invoice data is updated in the claims system electronically, and the payment is generated.

Because the claim form, estimate, and invoice are filed electronically, the exceptions that once occurred when a document was missing are eliminated. The records are maintained electronically in the imaging system. Because the electronic records are accepted as authentic, paper documents are destroyed. The filing of boxes of paper in the records center is eliminated.

4.4 Summary of benefits

This claim approval process example showed that, with Production Imaging Edition, you can improve client satisfaction, accelerate the business process, and reduce costs. Processes are streamlined and become more efficient and more accurate. Documents are not lost or misfiled. Transparency is improved, allows for more accurate control of the information, and offers opportunities to reduce fraud.

The following tasks are among those tasks that are improved or eliminated:

- ▶ Receiving documents, logging, counting, batching, date-stamping
- ▶ Sorting documents for filing and distribution
- ▶ Preparing file folders
- ▶ Filing documents
- ▶ Distributing documents for processing
- ▶ Photocopying for distribution
- ▶ Manually typing data
- ▶ Retrieving files from file cabinets
- ▶ Refiling documents and files
- ▶ Pending and suspending file management
- ▶ Searching for misplaced or lost files
- ▶ Purging files and removing selected documents for disposition
- ▶ Transporting documents to and from storage rooms or off-site storage
- ▶ Filing internal forms and copying correspondence
- ▶ Storage space-savings from eliminating file storage areas
- ▶ Archive filing costs on-site and off-site
- ▶ Reduced filing equipment costs
- ▶ Reduced copying costs

More information: For information about more potential cost saving areas, see Chapter 5, “Designing a production imaging system” on page 139.

Because the documents are entered into the system as they are received, every employee with appropriate access rights can view all relevant information anytime. You make better decisions faster because all the relevant information is available when they are needed. The system can reduce the clerical workload of the knowledge workers by automatically performing filing, routing, matching, and validation. Wherever possible, data entry is automated, and manual typing is eliminated. Data that was previously unavailable can be extracted and fed to analytics for improved decision making.

Chapter 5, “Designing a production imaging system” on page 139, explains how to design such a solution.



Designing a production imaging system

This chapter addresses production imaging system design using IBM Production Imaging Edition. The primary focus is on document capture, specifically in regard to the tools for handling discovery, requirements gathering, and functional design. This chapter also explores various alternatives with a look at the advantages and disadvantages of each alternative.

The other areas of the solution, primarily managing images and automating the business process, are equally important. Because these topics are explored in other IBM Redbooks publications, this chapter points you to resources that explore these areas in detail.

By the end of this chapter, we will have established the design for the sample application that we develop later in the book.

This chapter includes the following sections:

- ▶ Design goal of the production imaging system
- ▶ Capture system design
- ▶ Designing the capture for the auto claims scenario

5.1 Design goal of the production imaging system

Exploring potential production imaging projects begins with identifying the business goals of the business. At a high level, business goals can include the following examples:

- ▶ Reduce costs.
- ▶ Shorten the business process cycle time.
- ▶ Improve service.

To achieve these goals, we design a system that uses the document capture, imaging repository, and business process management aspects of a production imaging system.

As indicated previously, this chapter explores the details of the document capture design. To gain a deeper understanding of the document repository and business process management design, see the following IBM Redbooks publications:

- ▶ *IBM FileNet P8 Platform and Architecture*, SG24-7667
- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547
- ▶ *Introducing IBM FileNet Business Process Manager*, SG24-7509

If the current process uses paper documents, you can eliminate or improve many manual tasks, such as the following tasks, in the current business process to achieve the business goals:

- ▶ Receiving documents, logging, counting, batching, date-stamping
- ▶ Sorting documents for filing and distribution
- ▶ Preparing file folders
- ▶ Filing documents
- ▶ Distributing documents for processing
- ▶ Photocopying for distribution
- ▶ Manual typing of data
- ▶ Retrieving files from file cabinets
- ▶ Searching through files to find documents
- ▶ Matching documents against exceptions reports
- ▶ Refiling documents and files
- ▶ Pending and suspense file management
- ▶ Keeping calendars or diaries to track follow-up documents
- ▶ Searching for misplaced and lost files
- ▶ Reconstructing of lost files
- ▶ Purging files and removing selected documents for disposition
- ▶ Transporting documents to and from storage rooms or off-site storage
- ▶ Filing internal forms or copies of correspondence

In addition to eliminating and improving manual tasks, eliminating paper offers many other potential savings, such as the following examples:

- ▶ Storage-space savings from eliminating file storage areas
- ▶ Office space savings (including lighting, heating, furniture, and so on)
- ▶ Archive filing costs on-site and off-site
- ▶ Reduced workstation equipment and support costs
- ▶ Reduced filing equipment costs
- ▶ Reduced number of microfilm, cameras, processors, viewers, and consumables
- ▶ Reduced number photocopiers
- ▶ Reduced equipment maintenance of all types listed previously

5.2 Capture system design

You must consider the requirements that are unique to the area of capture, including the following requirements:

- ▶ Identifying how and where documents will be acquired, such as scanning, faxes, email messages, and imported documents
- ▶ Identifying the types of documents the application will process and the page types associated with each document type
- ▶ Deciding which data you want to capture from each page and which data might be manually typed or obtained by using database lookup
- ▶ Specifying the business rules that determine whether the captured data is valid
- ▶ Determining how to handle documents that are structurally invalid, pages that are not recognized, data that does not meet the business rules, or characters that are not recognized with high confidence
- ▶ Deciding how you want to export the data and documents at the end of the workflow

Before starting implementation, you must define the business requirements through collaboration with the various stakeholders. Initially this task involves examining the documents that you want to process, determining which fields you need to capture, and deciding what to do with the data and document after you capture it.

If you process various document types, you must decide whether the documents are presorted or processed as a mixed batch. If they are presorted, you might be able to simplify implementation by processing each type independently, with a separate application, workflow, or job for each type. However, if the documents are mixed batches, you need a more sophisticated system of page identification and document assembly.

Although the goal is to create a fully automated system, inevitably manual intervention is required at some points. The business requirements must specify how to determine if the information is accurate and how to handle exceptions with the data or the process.

At the early stage of a deployed capture system, it is common to review documents even when they passed validation at first to ensure that the system is doing what is expected in a production environment. As time goes by and more confidence is built in the new capture system, the validation process can be reduced to review only the pages with issues.

One way to look at the design is to consider three categories: document hierarchy, processing tasks, and capture workflow. The *document hierarchy* defines the structure of the content that we process. The *processing task* performs the work of the capture system, such as scanning and identifying pages and recognizing data. The *capture workflow* sequences the tasks into processes that handle the needs of different functional areas or input channels.

The *Taskmaster Application Development Guide*, SC19-3251, provides extensive tutorials for each of these areas. It is included in the Production Imaging Edition product documentation. Read this guide before you design and implement any capture system.

5.2.1 Document hierarchy

Taskmaster rules operate on batches, documents, pages, and fields. In Taskmaster, this structure is called the *Document Hierarchy* (DCO). The DCO is a core element of the design of the capture system. In addition to defining structure, the DCO provides the information that the system needs to assemble documents. It also enforces the integrity of the batches, documents, pages, and fields by using the information in the DCO. An application can have many DCOs to accommodate applications that require different classes of document structures.

The fictional company in this book, Fictional Insurance Company A, has an application with a simple document hierarchy. It has three document types: auto claims, estimates, and invoices. Within these document types, specific types of pages might only occur in a specific sequence. If the auto claim can have more

than one page, the company can define the first page as a unique page type. This way, the system can determine where an auto claim document begins and enforces integrity. When the pages are reordered, a document is flagged as invalid if the wrong type of page is set to page one.

Beneath the batch level, the document hierarchy defines the following information:

- ▶ The document types that the application can process. You may have only one type, or you might have multiple types. For example, the auto claims application processes claims, accident reports, invoices, estimates, and police reports document types.
- ▶ The page types within each document type. Each document might have only one page type, or it might have multiple types. The accident report document type includes three types of pages: front page, diagram page, and signature page.
- ▶ The number and order of pages within each document type. Pages can be required or optional. The accident report document has three pages. An invoice can contain one or many pages.
- ▶ The data fields within each page type. Data fields can also be required or optional. The invoice document has single fields such as invoice number, invoice date, and vendor number. It also includes repeating line items with fields such as item number, item description, quantity, unit cost, and total cost.

In this scenario, after gathering information about the document types and their properties, we design a document hierarchy and enter it into the system by using Datacap Studio. For details, see Chapter 6, “Implementing the capture solution” on page 179.

5.2.2 Capture processing tasks

In many instances, documents are acquired as a stream of pages where little information is known about the structure or content of the pages. Initially, the type of document and the processes that need to occur to correctly handle the document are unknown. For example, when documents are scanned, the input to the capture system might only provide a series of image files and the type of batch. The job of the capture system is to make sense of the images and perform a series of tasks that process them appropriately.

What tasks are involved in the capture process? Typically the tasks include extracting useful data from the input, validating the input, formatting documents, and outputting the data and documents to business systems. Poorly scanned images and documents of various types that require human reviews might be

mixed together. These factors introduce exceptions and variation that need to be detected and processed effectively.

The capture process performs the following essential elements, which are incorporated into the capture system design almost always in the order shown:

1. Document acquisition

Documents are input into the system by scanning, faxing, importing, emailing, or web services.

2. Image enhancement

Images can be enhanced to improve recognition and readability and to reduce file size. This enhancement can be done at a scanner by using the built-in capabilities of the hardware or driver. Alternatively, it can be done by using the Taskmaster image enhancement features.

3. Page identification

The type of each page must be identified (classified), automatically or manually. For example, a barcode can be used to automatically identify a page. A document often consists of a specific type of leading page followed by one or many trailing pages.

4. Document assembly

The capture assembles multiple images into documents where a single scanned batch or fax transmission can contain multiple documents. Such information as the page types, number of pages, and order of the pages provides the basis for automating document assembly. The document type is typically determined automatically by using the document creation function.

5. Recognition

Recognition includes using Optical Character Recognition (OCR), Intelligent Character Recognition (ICR), Optical Mark Recognition (OMR), barcode recognition, or database lookups to lift data and supplement the data with additional information.

6. Fingerprinting

Fingerprinting is commonly used to differentiate between multiple formats of the same page type. Fingerprinting matches the best variation on a page type and captures the offset that is needed to adjust an image for locating data accurately.

7. Locating data

Data in text on a page can be located in zones, by using keyword searches, through regular expressions.

8. Validation

Extracting data by using any of the recognition methods has inherent limitations for many reasons. Examples of such reasons include a damaged source document, poorly scanned document, poorly printed document, and inaccurately entered data. Validation of the data is essential to obtain accurate results by using such techniques as check digits, length checks, format checks, cross-totaling calculations, value comparison, and data lookups.

9. Routing

When exceptions occur, routing is used to queue batches or documents for exception handling. For example, a document that is missing a page or poorly captured might need to be fixed at a scanning workstation.

10. Verification

Often a design goal of the capture system is to reduce or eliminate manual verification. However, when low confidence results or validation errors exist, they might need to be handled by human operators. Correct results need to be confirmed, and errors need to be resolved. Verification can include typing from recognition, typing from image when recognition is not used, and typing from documents when the documents are not digitized.

11. Export

The system transfers documents and the data to external systems (for example, IBM FileNet Content Manager) where they are stored and processed by the business. Extracted data is exported to XML files or databases to update applications.

Sequence of the design elements: Although most capture system design follows the previous order, the process can be done in multiple ways. For example, you can use fingerprinting (step 6 on page 144) as page identification (step 3 on page 144) before recognition (step 5 on page 144). Although fingerprinting is *not* a preferred way of page identification, it can be used.

Page identification: Multiple methods of page identification (PageID) are possible. Using fingerprinting as mentioned previously is one such method.

With Taskmaster, these elements are implemented as rules. Rules are run by the Taskmaster Rulerunner service. This method provides a flexible way to implement all of the variations and exceptions that are seen when capturing content in a scanned or document format.

In this scenario, many processing tasks and rules are already defined in Taskmaster. We only need to adjust these tasks to meet the specific document structure. Taskmaster unifies the tasks definition with the document hierarchy.

We configure rules and tasks with the same tool, Datacap Studio. For more information about using Datacap to configure rules and tasks, see Chapter 6, “Implementing the capture solution” on page 179.

5.2.3 Capture workflow

During the data capture process, documents go through a workflow that consists of several tasks. Some tasks require operator intervention, where others run automatically. A workflow job consists of a series of tasks and defines a way to process a particular batch of documents. Because the tasks can be reused in multiple jobs, we can add as many jobs as we need to handle the processing scenarios in this book. The design must include workflow jobs that specify and execute the capture process.

For example, we might have several input channels, such as scan, fax, and email, for the same types of documents. We can construct three workflow jobs, one for each input channel, and have each job share tasks for recognition, data extraction, and export.

In addition to defining the process flow, the workflow also implements functional security that so that we can determine who can access the work in progress and who can perform specific tasks with particular types of documents. For example, processing claims for motorcycles might be similar to handling claims for automobiles. However, the people who verify the documents might be in a different department. A separate workflow might be used to accommodate this difference.

For a definition of the capture workflow for the scenario in this book, see Chapter 6, “Implementing the capture solution” on page 179.

5.2.4 Capture design considerations

This section highlights the areas to consider when designing a capture system and indicates the alternatives that are available. You can select from multiple options depending on your business and technical requirements.

Document acquisition

Taskmaster services various input channels that deliver documents in several formats. Channels or methods of capturing documents include scan, fax, email, file import, and a web service. Some of the considerations for each channel are provided in the following sections.

Scanning

Direct scanning is typically done by internal users. Both thick and thin client options are available.

Thick client scanning is used in centralized scanning operations that use mid-range to high-volume scanners that have heavy-duty cycles. These types of scanners support continuous operation in multiple shifts. Even though a lower-cost scanner might have a high scan rate, it might not be designed for continuous operation.

The Taskmaster scanning user interface is production-oriented to support highly efficient operation of the scanner. In this environment, scanners are operated nonstop. Scan operators occasionally check the scan quality of images. The goal is to maximize the throughput of the production-level scanners.

Thin client scanning can also be used in centralized scanning operations but is more commonly deployed for distributed capture. Common deployment models are dedicated scanning stations connected to mid volume production scanners and user workstations connected to low volume scanners.

Multifunction devices (MFD) have integrated scanner, printer, copier, and fax capabilities. Production-level MFDs can be operated as stand-alone devices without being connected to a workstation. In this mode, the MFD control panel is used to control the scanner. Images can be transferred to a well-defined storage location by using the network filing, File Transfer Protocol (FTP), or email functions of the device. Taskmaster can import and process the documents by using its virtual scanning and email import actions.

One area of common confusion is the difference between thin client scanning and operating an MFD directly. If you scan with an MFD by using thin client scanning, the MFD is connected to a workstation by using a TWAIN driver and the web user interface on the workstation provides the scanning control panel. This method is used with lower-end desktop MFDs and is not used with higher-end production MFDs.

Consider the following additional factors:

- ▶ Many current generation devices include image enhancement features that are run within the scanner hardware or in the scanner driver. In either case, the resulting image might have improved readability, improved recognition results, and reduced file size.
- ▶ Scanners are available for specialized purposes, such as remittance scanning and large format document scanning. These devices might not have Image and Scanner Interface Specification (ISIS) or TWAIN drivers. Therefore, they interface with Taskmaster by using import features.
- ▶ If you expect an MFD to be used full-time as a scanning device, consider using a dedicated scanner instead. Production scanning can handle larger scan jobs that might occupy an MFD that needs to be shared by a workgroup.

Fax

Production Imaging Edition software works with fax server products so that documents that are sent to a fax server can be imported into the capture system and processed in the same manner as scanned documents. The Production Imaging Edition software does not include a fax server.

Fax is typically used by external users. The trend in many organizations is to reduce the internal use of fax for capturing documents. This trend is due to the lower quality of the image and the greater time needed to send a fax compared to remote scanning. However, because fax requires low bandwidth, its use is common in situations where only dial-up connections are feasible.

The primary disadvantage of fax is low image quality. The quality of the equipment varies resulting in inconsistent image quality. Fax image resolution is low. *Standard mode* provides a horizontal scan at 200 or 204 scan lines per inch. It provides a vertical scan at 100 or 98 scan lines per inch. *Fine mode* provides a horizontal scan at 200 or 204 scan lines per inch. It provides a vertical scan at 200 or 196 scan lines per inch.

Each fax transmission is received as a single TIFF or PDF file that contains multiple images. Taskmaster can burst the file into individual image pages for processing by the system. The image enhancement actions improve the ability of the system to recognize text. Taskmaster can normalize the dimensions of the image so that all the images are 200 dpi in both dimensions. It can also compress images to the TIFF Group 4 format.

Email

Taskmaster can capture and process email messages and their attached files. In addition to scanned images, Taskmaster can accept various electronic formats, such as word-processing documents and spreadsheets. Electronic documents can be converted to TIFF by Taskmaster so that they can be processed as images for data extraction and export.

Consider the following common scenarios for using email:

- ▶ Trailing documents can be received directly from customers or other external parties. In the scenario in this book about the insurance company, customers who register a claim can be allowed to send supporting documents by email to a service email account.
- ▶ Email can be used as a replacement for fax as a way to transmit scanned or electronic documents.
- ▶ Email can be used to interface with MFDs.

File import

File import is a common method for inputting files into the system. The virtual scan (VScan) features of Taskmaster are used to import files. File import can be done in an attended or unattended mode. In an attended mode, a user starts the virtual scan by using the thin- or thick-client user interface. In an unattended mode, the virtual scan is run by the Rulerunner service, which runs as a Microsoft Windows service.

Consider the following common scenarios for using file import:

- ▶ Receiving images from an external party. For example, a financial institution might receive loan file images as part of the process for purchasing loans from another financial institution.
- ▶ Receiving images scanned by a scanning service. For example, large quantities of documents might be scanned by a third-party service as part of a backfile conversion.
- ▶ Interfacing with fax or MFDs.
- ▶ Interfacing with a scanner that does not have a TWAIN or ISIS driver. Some specialized scanners operate in this fashion.

Web service

Taskmaster exposes the document processing capabilities as a web service. The web service can run the background document processing tasks. This method is used by software applications that need to process documents.

Consider the following common scenarios for using a web service:

- ▶ Processing previously scanned and stored documents that were not previously processed for recognition. A bank that stored loan documents when a loan was originated might want to perform data extraction on the same documents years later when a loan is modified.
- ▶ Providing a service where documents can be processed in an ad hoc manner. An organization might provide a service to upload documents for recognition and transformation through a web application or portal.

Centralized capture

With centralized capture, dedicated staff and equipment process documents in a factory-like setting. Documents are mailed or delivered to the central location where documents are prepared into controlled batches. Batches are scanned on high-speed scanners. Other tasks, such as indexing, data entry, and fixup, are performed on separate workstations so that each task is optimized and labor and other resources are used efficiently at the central location.

Centralized capture offers the following advantages:

- ▶ Economy of scale
- ▶ Standardized processes
- ▶ Dedicated trained personnel who only do capture-related tasks
- ▶ Easier to maintain image quality controls
- ▶ Availability of original documents to verify authenticity

Centralized capture has the following disadvantage:

- ▶ Documents must be delivered to a central location.
- ▶ Users understand less about the documents.
- ▶ Corrections might require returning documents to the sender or interacting with remote users to correct problems.

Decentralized capture

With decentralized capture, remote offices or individuals scan, fax, and process documents, but they do not send the paper to a central location. Staff is not dedicated to performing capture activities. Capture might be done directly by the customer or by an external business partner.

Decentralized capture has the following advantages:

- ▶ Documents do not need to be mailed or shipped to a central location.
- ▶ Documents are stored into the repository more quickly.

- ▶ Users can correct errors immediately.
- ▶ Users understand the documents and can more accurately enter and correct data.
- ▶ Work can be offloaded to a partner or customer by using self-service.

Decentralized capture has the following disadvantages:

- ▶ Equipment is needed at each location.
- ▶ It is harder to maintain standardized processes.
- ▶ More users need to be trained.
- ▶ Users do not perform capture functions all the time and, therefore, do not handle the tasks as efficiently.
- ▶ Image quality varies, and image quality issues are more difficult to correct.
- ▶ Authenticity is more difficult to verify.

In many instances, organizations use a blend of these models. The capture system needs to accommodate the constraints and demands of the business. Organizations have multiple applications that require one, the other, or both models. Fortunately, in this scenario, the production imaging system accommodates both models, so that we can decide which method works best for each application.

We must also consider the network capacity to determine if it is sufficient to handle the required load. In some locations with low bandwidth, networks might need additional bandwidth to accommodate higher volumes of imaging network traffic.

In either scenario, the background processing of documents is handled centrally using the Rulerunner service. Background processing includes image enhancement, OCR or ICR, format conversion from input or for export, and export. Because these are processor intensive activities, they are handled most effectively in servers or high-end workstations. In this manner, client workstations do not need to have software installed to perform these functions.

In the scenario in this book, we use centralized and decentralized capture. Policy holders can send signed claim forms and follow-up documents remotely by using email or fax directly to the capture system. Repair shops can submit invoices by using fax or email. In both cases, the documents are verified and processed centrally. Regional offices and agents capture paper documents by using web-based scanners and MFDs to scan and verify the documents remotely by using Taskmaster Web. Each party can still choose to send paper documents to the current capture center for central processing.

Image enhancement

Images can be enhanced to improve recognition and readability and to reduce file size. Image enhancement is most important when using OCR and ICR or to improve the format of faxed images. Taskmaster includes image enhancement capabilities for this purpose. The current generation of document scanners often includes image enhancement capabilities in the hardware or scan driver that can be configured in the scanning user interface. Use the capabilities of your scanning hardware for image enhancement, and supplement those capabilities with the Taskmaster enhancement features.

Page ID and document assembly

Page ID and document assembly are often referred to as *classification*. This process identifies the type of each page in a batch and creates documents from the stream of pages. *Page identification* is the process of identifying the type of each page. *Document assembly* is the process of determining where each document begins and ends.

Orchestrated classification

Taskmaster performs automated classification by using the *orchestrated classification technique*. Orchestrated classification uses page identification rules, document integrity rules, and document creation rules to automate the classification process. Classification can also be done manually in a scanning or verification user interface.

Orchestrated classification uses a set of rules that takes a stream of pages. Then it optionally enhances images, identifies each type of page using one of many methods, creates documents from the pages, and validates the resulting structure. All the classification processing can occur in a single module in one workflow step. If necessary, you can have multiple types of classification modules. Classification can use any of the processing actions in Taskmaster.

Page identification

Documents are created and separated based on the page types and a set of document integrity rules. Pages can be determined by one of the following methods:

- ▶ Barcode
- ▶ Pattern match using image anchors
- ▶ Pattern match using text anchors
- ▶ Match image-based fingerprint
- ▶ Match text-based fingerprint
- ▶ Match regular expressions to recognized text
- ▶ Document structure using rules

Consider using barcodes as the primary method of page identification for forms that you control. When you do not control the layout of the form, you can use the other page identification methods depending on the characteristics of the pages.

Document assembly

In Taskmaster, the system determines document separation and document type by matching the document hierarchy to the identified pages. After pages are identified, Taskmaster uses the information in the document hierarchy to determine the correct document type. For example, a page of type “Accident Diagram Page” is part of an “Accident Report” document, where a page of type “Auto Claim Page” is part of an “Auto Claim Form” document.

Each page has the following variables that define the structure of the parent document:

- ▶ Maximum number of pages of this type for each document (0 means no maximum)
- ▶ Minimum number of pages of this type for each document (0 means no minimum)
- ▶ Order, which is the position of this page relative to other pages in the same document (0 means any position)

Taskmaster uses the information in the document hierarchy to assemble individual pages into multipage documents.

A common approach to document separation is to use barcode sheets between each document or printing barcodes on the first page of a document. During scanning, several documents of varying lengths can be scanned in a batch, with barcoded sheets separating each individual document. The system saves the documents as separate documents automatically. Barcoding can also be used to identify the type of document or pages.

Position barcodes vertically. Keep in mind that scanners and fax machines can produce vertical lines on a page when dirt is on the scanning sensor. If the line is parallel to the barcode line, it can make a barcode unreadable. If the line runs perpendicular across the barcode, it is readable.

Recognition, fingerprinting, and locating data

Recognition is used to read data from images by using OCR, ICR, OMR, or barcode technology. Recognition is used in two primary use cases: to automate document indexing or to reduce data entry typing.

Indexing is the process of identifying the documents stored in FileNet Content Manager. Documents are identified with properties that are stored in a content engine catalog. The process of entering these properties is called *indexing*.

Users search for documents by using these properties. As a result, these properties must clearly identify each document with information, such as claim number, invoice number, customer ID, and vendor name. Usually only a few properties are used to index a document.

Data entry is the process of typing data into a database or application system. Documents can contain dozens or hundreds of fields of data on many pages. In a manual process, users type data by looking at the paper document or at an image of the document in a window. Typing from a window is called “type from image”.

When we design the capture system in the scenario in this book, we use recognition features to read the data from images so that we can reduce the amount of manual typing.

Data recognition and extraction can be highly accurate when certain conditions are met. An understanding of the document characteristic is vital because you need to choose the most appropriate techniques or combination of techniques that are most effective on the types of documents that you have.

Documents come in two classes: document that you control and documents that you do not control. Documents that you control are often internally generated documents that can be redesigned to make recognition more effective. If a document is not designed for recognition, it can be more difficult to process and can have lower confidence from the recognition process. However, documents that are outside of your control generally cannot be redesigned, and you must accommodate the existing format.

When you control the layout of your forms, a good practice is to redesign forms to improve recognition. In some cases, this redesign might be necessary to achieve high confidence recognition results. Some forms might need minor changes to improve results, but others might need extensive redesign. Engaging a form design expert is one way to achieve the best design for your documents.

The following factors can improve results:

- ▶ Use barcodes to identify document types and prepopulate a document with data. For example, internally generated documents can be printed with barcodes on them to identify the document type and indexing data. When they are returned, you automatically recognize the document type and indexing data rather than manually type the data. The business process capabilities have features that match the document to a pending task that is waiting for the document to arrive. Using barcodes in this way is included in our use-case application scenario.
- ▶ Use machine print whenever possible. Forms that are completed and printed online generally contain printing that is easy to extract. Prefilled data of printed forms can also be easy to extract.

- ▶ Use hand printing only in controlled circumstances. Handprinted information must be printed in boxes or other guides that show the user where to enter each individual character. Other types of handprint require specialized software beyond the scope of this publication.
- ▶ Clearly identify data locations by using unambiguous prompts or by using specific zones on the page.
- ▶ Include multiple fields that cross-check the data, or use data that is designed to self-verify by using check digits.

For additional factors, see Chapter 9, “Best practices and recommendations” on page 289.

Fingerprinting

Within a specific document type, often many variations can exist in the format and layout of the printed information on the page. The existence of variations does not refer to minor shifts in position on a page. Instead, it refers to the wider variations from a different version of a form or from documents that are created by outside parties where you cannot control the layout. In our design in the scenario, we must determine which method to use to handle these variations.

Fingerprinting is a technique that Taskmaster uses to differentiate between different multiple formats of the same page type. Fingerprinting matches the best variation on a page type and captures the offset needed to adjust an image to locate data accurately.

For highly structured documents, you can also use image or text-based anchor fields. These marks are on the page, in specific positions. This approach is effective for fixed-format forms where you have control of the forms design.

If you do not use fingerprinting or anchors, you can still deal with the format variation by using keyword searches and regular expressions to find data within the full text of a page.

In the design, we must examine all of the different types of documents and decide which approach is more effective. The rule of thumb is that, if you can hold two different pages 10 feet away and see which page is different, you can use fingerprinting to identify the layout.

Location techniques

Data in text on a page can be located in zones or by using keyword searches and regular expressions. These techniques can be used in combination to handle forms that have both fixed and variable data locations.

If you use fingerprints or anchors, then you can accurately register the location of zoned fields. Zones can be prepared in advance or dynamically. The most

flexible option is *Intellocate*, which allows Taskmaster to learn page layouts from users. It uses a hybrid approach that combines both zones and text searching.

When we design a system, we must examine the individual documents to determine which location technique can be used for the data on our documents

Intellocate

Intellocate is technology that allows Taskmaster applications to learn. Location rules are used to automatically locate some of the data from these documents by using keyword searches or regular expressions. Information that cannot be automatically found by using Intellocate can be identified and captured quickly and easily by a verify operator using the Click'n'Key capability.

With Click'n'Key, the operator clicks the words on the image, and the data is entered into the data field. Behind the scenes, the system remembers the locations where the user clicked. When this task is complete, Intellocate saves the zones for the fingerprint. Then, the next time a similar document is encountered, the fingerprint is matched, and all of the data is read by the zones.

For more information about Intellocate techniques, see Chapter 10, “Dynamic technologies” on page 305.

Data validation

The purpose of validation is to determine whether captured data conforms to specified business rules, as in the following examples:

- ▶ Does an expense lie within permitted limits?
- ▶ Are dates valid and within a permitted range?
- ▶ Is the total cost calculated correctly?
- ▶ Does the vendor information match the information stored in a database of approved vendors?
- ▶ Does a field value match one of a set of permitted values?

Taskmaster performs validation by using rules that you create and attach to specific items in the document hierarchy. For example, to check whether an expense lies within permitted limits, you might first create a rule that performs the following tasks:

- ▶ Ensures that the expense field contains numeric data in a valid currency format
- ▶ Determines if the value is less than or equal to the maximum permitted limit
- ▶ Performs exception handling if the value is invalid or higher than the permitted limit

OCR and ICR read what the user entered. The user can still write or print invalid data on the document. The scope of the capture process usually includes validating that data was correctly read from the page and that the content of the data is valid.

Validation can include simple field-level checks, field cross-checking, and lookups to external data sources. At the field level, it checks the ranges of values, valid format (for example, dates), check digits, choice lists, and so on. Cross-checking can include totaling columns and checking against total amounts, such as checking the line item detail total against an invoice total field. Taskmaster can query database tables to look up valid values. Lookups are used to check account numbers, product codes, and other sorts of master data.

Depending on the business use of the data, the application might need absolute data accuracy, but sometimes applications might want to accept lower confidence data.

For example, if you are processing financial transaction, it is likely that you want data to be accurate. However, if you are processing survey cards, you might want to accept incomplete responses of lower confidence because you are more interested in the receiving as many responses as possible.

For each type of page and for each field, you must determine the level of confidence that is flagged by the system and displayed to an operator for verification.

Validation is run in the background task after recognition and at the Verify task. Data that does not conform to business rules is flagged. Documents that do not conform to business rules are routed for verification by using workflow. Data can be flagged down to the character level. Validation is also executed from the verification user interface when a user types corrections.

Routing

Workflow routes exceptions for manual verification. You can route an entire batch, or you can split batches so that problem documents are handled in a separate batch from the good documents.

In the insurance company scenario, we must determine the types of exceptions that we will handle and who will handle them. Common types of exceptions include rescanning, page identification, data verification, and data exceptions.

Verification

During verification, an operator views data entry panels and document pages for manual checking, for possible correction, and to type data. You want to display pages to an operator when one of the following primary conditions exists:

- ▶ The batch failed document integrity checking during document assembly.
- ▶ A page contains one or more characters or OMR fields that were marked “low confidence” by the recognition engine.
- ▶ A validation rule failed, indicating that the data does not conform to business rules.
- ▶ The application does not recognize that data and verification screens are being used to type data from the image or the document, which can be for indexing or data entry purposes.

When a batch fails document integrity checking during document assembly, you can have a user manually identify pages, by using a special verification task called *Flex ID*. Flex ID handles the manual page identification, and you display the thumbnail images of the page. Then the user rearranges the thumbnails and selects the page type for unidentified pages.

The other conditions require the user to enter or correct data. Several thin- and thick-client verification user interface options are available. All of these options display the image, the data fields, and snippets of images where data is on the image page.

Take a single-pass approach to verification. Some other systems promote a two-pass approach where individual character-level corrections are handled in the first pass, and in a second pass, field-level corrections are made. Our experience is that a single-pass approach is more efficient. The user interface has keyboard shortcuts that navigate efficiently at the character level, making a separate first pass unnecessary.

You can control what is displayed to the user in the design and configuration. As part of the design in this scenario, we decide what level of verification is needed. Many options are available. For example, we can display every document and page, only pages where we have data, the first page of a document, only documents and pages with exceptions, and pages that do not conform to business rules. This setting is a business decision and varies depending on factors such as the types of documents, business controls, or the comfort level of the user with automating the process.

Multipass verification

To satisfy business requirements, you can consider whether you want more than one person to verify the data. Multipass verification can display the same page to multiple operators to ensure accurate data entry and verification. In some cases, the business financial controls require a separation of duties that requires more than one user to enter or validate specific data fields.

Taskmaster supports two main implementations of multipass verification: two pass and double blind. Other implementations are possible, but this book focuses on these two that are supported by the standard user interfaces.

In two-pass verification, the following process occurs:

1. An operator (or a recognition engine) enters the initial value for each field.
2. Taskmaster displays the page to a second operator but hides the initial values. The operator enters a new value for each field. If using a recognition engine to implement the first pass, you might choose to show only low confidence fields to the operator.
3. For each field, Taskmaster compares the new value to the initial value. If the values match, Taskmaster accepts the value. Otherwise, the operator must re-enter the value. Taskmaster accept the value only after the operator enters the same value two times consecutively.

In double-blind verification, the following process occurs:

1. An operator (or a recognition engine) enters the initial data values.
2. Taskmaster displays the page to a second operator but hides the initial values. The operator enters a new value for each field, and Taskmaster saves all the values (no comparing).
3. Taskmaster displays the page to a third operator. The operator can see both the initial value and the second value.
4. For fields where the initial value and the second value are different, the operator must select which value is correct or enter a new value. If entering a new value, the operator must enter the same value two times consecutively.

Web-based verification options

Taskmaster includes several different user interfaces that have different design features. In the insurance company scenario, we choose the user interface or combination of user interfaces that meet our requirements. For information about the detailed features of each interface, see the *Taskmaster Application Development Guide*, SC19-3251.

Table 5-1 summarizes the key features.

Table 5-1 Features of various user interfaces

Design features	Task ID
Verify recognition, custom panels, batch restructuring, and two pass	prelayout.aspx
Verify recognition, custom panels, and line item details	averify.aspx
Verify recognition and overlay data entry fields on the image	imgEnter.aspx
Key-from image, manual page identification, manual registration, two pass, and double blind	aindex.aspx
Manual page identification and fixup with thumbnails	ProtoID.aspx

Fixup tasks

Fixup activities adjust and enhance pages, move pages within the batch, reconstitute documents, and reorganize the batch.

If a workstation has a scanner attached, the fixup can also rescan. *Rescan* is a physical process that scans one or more pages and replaces existing image files with the new files. One constraint for rescanning is that the workstation where rescanning occurs must have access to the physical documents.

In centralized operations, scanned documents are often stored in boxes on-site for a short time when rescanning occurs. In decentralized environments, batches must be routed to the person who scanned the document.

Many customers find it is more efficient to rescan an entire batch rather than to pull out the individual document and rescanning the individual page. This preference depends on the application.

Sometimes batches are sent to a fixup task to delete documents from a batch. For example, you might need to send the original document back to the sender if it is not a valid document. In this case, an operator must pull the original paper document from the physical batch and send it (perhaps by mail) to the originator. The images can be flagged for deletion from the batch.

Export

Taskmaster supports data and document export.

Data export

Taskmaster can export data to a text file, an XML file, or a database. This choice depends on the means of interfaces that are available in the target application. These three methods are equally easy to configure. You can export multiple formats for the same batch.

Document export

Document export creates documents from the scanned pages and stores the documents in one or more systems. The Production Imaging Edition component for storing document is FileNet Content Manager.

In addition, the Taskmaster software can export to a file system, other IBM Enterprise Content Management (ECM) system, third-party ECM systems, and collaborative systems including Microsoft SharePoint. When a batch is exported, the destination for each document is determined at the document level. Therefore, documents in the same batch can be stored in multiple systems.

When designing the system, you must determine the output system, the file format, and the document properties of the exported documents.

Considerations for exporting file formats

The primary output options are TIFF, PDF, and PDF/A. Documents are processed as individual TIFF image pages, but they can be converted to different formats for export. For example, scanned or faxed images in TIFF format can be converted to PDF/A for export.

In addition, you might consider the following export options:

- ▶ Documents can be exported in their original format. For example, when a customer sends a Word document by email, the document is processed by using the image functions as TIFF pages. When the export is done, the original Word document can be exported.
- ▶ Documents can be exported in multiple formats. In the same example, the Word document can also be rendered in PDF/A format and stored for archival purposes. It can also be stored in the original Word format.
- ▶ Images can be redacted, where specific area of the image is erased or obscured. Redaction can be used to cover ID numbers, credit card numbers, or other protected information.
- ▶ Additional capabilities are available for conversion to other formats including a rich text format (RTF), HTML, Excel, Word, and various text formats. These

formats are used for specialized applications and are secondary options for imaging applications.

Considerations for exporting document properties

When documents are stored in the repository, you store the document and catalog the document by using a small set of document properties, such as the claim number, policy number, customer name, or invoice number. Now you use the data collected throughout the capture process on the batches, documents, and pages. You store selected data fields with the documents into the FileNet Content Manager repository.

In FileNet Content Manager, each document belongs to a document class where the document class specifies a list of document properties. These classes are mapped to corresponding Taskmaster document types, in a one-to-one or one-to-many relationship. The properties of the document class map to Taskmaster fields and variables on the batches, documents, or pages.

A unique feature of Taskmaster is the ability to update the properties of an existing document in the FileNet system, not just the document you are exporting. For example, the document you are exporting is a change-of-address request, and a field contains an updated postal code. In this case, you can update the postal code on other documents that are already stored in the FileNet system.

You can also use this update feature to implement an early export scenario. In this case, documents are exported before the entire recognition and verification workflow job is completed. With this scenario, documents are exported to be stored under document management system, even though all the steps of the workflow have not completed. In a later workflow task, additional data can update the document properties.

Exporting to the FileNet Business Process Manager process

After a document is stored in the FileNet Content Manager repository, both the document and the data are available to a FileNet Business Process Manager (BPM) process. The act of adding the document triggers an event to FileNet BPM, initiating a new BPM process or updating a currently running BPM process. Any data in the document properties, and the document itself, can be included in the BPM process.

With existing BPM processes where the process steps include data entry, consider shifting the data entry function to the capture system. With the data recognition and extractions capabilities of the Taskmaster, these functions can be implemented so that they require less manual typing of the data.

5.2.5 Discovering the capture process

Every organization has a different starting point with different requirements. Some organizations already have implemented a capture process and are updating their system to take advantage of advanced capabilities. Others are implementing capture for the first time or are using Taskmaster for the first time on a new application. Some want to scan documents and store them in an ECM repository. However, others want to extract data from documents for updating business systems. Yet others want to focus on classifying documents and reducing manual paper document preparation. In any case, understanding the capture process is a key step in the design process.

As a starting point, in the insurance scenario, we must define the business requirements through collaboration with the various stakeholders. The process of gathering required information about a business problem is called a *walkthrough*. In a walkthrough, you learn about the current sources and methods of handling documents, and examine the documents. You learn the characteristic of the documents, how they need to be processed and stored, and determine which fields need to be captured and what to do with the data after you capture it.

In addition to the mechanics of configuring the capture application, you must consider the physical aspects of the paper handling tasks, such as whether document types are mixed together or presorted. If they are presorted, you might be able to simplify implementation by processing each type independently. If you process mixed batches, you can automate and reduce the amount of manual sorting using the orchestrated classification techniques.

If data or index values are manually typed, you can examine the characteristic of the data printed or written on the documents and determine if the data can be extracted by the software. Alternatively, you can propose a redesign of the document layout or forms so that the data can be captured more effectively.

Although the goal is to create a fully automated system, at certain points, manual intervention is required. The business requirements must specify how to determine if the information is accurate and what you will do if a problem arises.

Because this Redbooks publication is an introductory guide, it does not provide a detailed methodology for determining business requirements. Instead, it provides guidance about the key information that you need to gather and review.

5.3 Requirements gathering

This section lists the questions that help to identify the application requirements and the relevant details to design the capture system. The information that you discover includes the following categories:

- ▶ Current capture or document processing environment
- ▶ Physical locations that receive and process documents
- ▶ Types of documents, their characteristics, and the data they contain
- ▶ Business rules that validate whether the data is valid
- ▶ Document volumes and time constraints
- ▶ Business requirements for dealing with exceptions
- ▶ Output requirements for data and documents
- ▶ Scanner requirements
- ▶ Hardware and software requirements

5.3.1 Requirements for current capture or document processing environment

With this category, you discover the characteristics and details of the business processes and systems that are currently in place. You look for the specific tasks that are performed, the sequence of those tasks, and the overall time it takes from document arrival through completion.

Scanning

If the current process involves scanning documents, you must identify the current systems and methods. You can consider redesign of the existing processes in light of the capabilities of the Production Imaging Edition system compared to existing methods. You can evaluate potential reuse of existing processes, equipment, and systems.

Identify the scanning requirements:

- ▶ Are paper documents currently being scanned?
- ▶ At what point in the business process are they scanned: upon arrival, in the middle of the process, at the end of the process, or a mixture?
- ▶ What equipment and software are being used to scan the documents?
- ▶ Will the current equipment be replaced, or will it be used with the new system?
- ▶ Can the scanners handle the projected peak volumes based on comparing the scanner specifications to the scan volume?
- ▶ Will the scanner handle de-skewing and noise removal?

- ▶ For each location, will scanning be done by using thick-client ISIS or thin-client TWAIN scanner drivers?
The preferred practice is to test a specific scanner interface, driver, and scan hardware in a test environment.
- ▶ What happens to the paper documents after they are processed? Are they stored on-site, returned, stored off-site, or destroyed?
- ▶ Will the new system change the way paper documents are handled after they are scanned?

Processing

Processing paper documents is a labor-intensive operation. By explicitly documenting the current processes, you can identify the specific areas of process improvement.

Identify the processing requirements:

- ▶ How many people “touch” the document from arrival to completion, and in which departments or locations do these people work?
- ▶ What is the current document handling process?
- ▶ Are documents processed centrally or at remote locations?
- ▶ How many people are involved in processing documents?
- ▶ Which processing is currently being performed?
 - Receiving documents, logging, counting, batching, and date-stamping
 - Sorting documents for filing and distribution
 - Preparing file folders
 - Filing documents
 - Distributing files or documents for processing
 - Photocopying for distribution
 - Manual typing of data
 - Retrieving files from file cabinets
 - Searching through files to find documents
 - Matching documents against exceptions reports
 - Refiling documents and files
 - Pending or suspense file management
 - Keeping calendars or diaries to track follow-up documents
 - Searching for misplaced or lost files
 - Reconstructing lost files
 - Purging files and removing selected documents for disposition
 - Transporting documents to and from storage rooms or off-site storage
 - Filing internal forms or copies of correspondence

Policies and systems currently in place

Identify the policies and systems that are currently in place:

- ▶ Has our organization approved the destruction of original paper documents following scanning?
- ▶ What systems are used for tracking and inventory of paper documents and files?
- ▶ What ECM or other systems are currently involved in the current scanning or capture operation?

Time frames

Identify the requirements regarding time constraints:

- ▶ How long does it take for a document to be processed from arrival to completion?
- ▶ Are there significant differences in time depending on document type? If yes, identify the differences.
- ▶ What steps in the process take longer than desired?

5.3.2 Processing location requirements

This section identifies where documents originate and how to gather them for processing.

Physical documents

Identify the requirements for physical documents:

- ▶ How many physical locations create or receive physical documents?
- ▶ Are the physical documents processed in the location where they are received, or are they moved to a central location for processing?
 - How are they moved: by mail, internal courier, or external courier?
 - Are photo or scanned copies made before they are moved?

Electronic documents

Identify the requirements for electronic documents:

- ▶ How many physical locations create or receive electronic documents?
- ▶ Are the electronic documents processed in the location where they are received, or are they moved to a central location for processing?
 - How are they moved: by email, electronic media, file copying, or file transfer?
 - Are copies made before they are moved?

5.3.3 Document type requirements

The questions in this section help to identify the documents types, how they are created, and their characteristics. You must identify and gather single and multiple page samples of all document types.

Identify the requirements for document type:

- ▶ What are the document types and any subtypes, that we process? Consider the following examples:
 - Packing slips for complete, partial, back ordered shipments
 - Invoices, including purchase order invoices, non-purchase order invoices, preapproved invoices, trade invoices, non-trade invoices, and credit memos
 - Attachments, including shipping confirmation notices and acknowledgement of receipt forms
 - Loan applications, including the application form type by form number
 - Insurance claim, such as the claim form by form number
 - Tax forms, including the form number and year
- ▶ Who creates the documents?
- ▶ Can the design of the documents be changed if necessary to increase recognition accuracy?
- ▶ If documents are created by external parties, approximately how many sources are involved?
- ▶ What is the input source for each type of document: scanner, fax, email, or other systems?
- ▶ For each type of document, does it have a fixed number of pages or a variable number of pages?
- ▶ What is the number of pages per document?
- ▶ For images, what is the image resolution and format (black and white, color, gray scale)?
- ▶ What is the input file format for electronic documents?
- ▶ Do documents contain more than one business transaction?
- ▶ Do people stamp, mark up, or write on documents as they are processed?

5.3.4 Captured data requirements

Whether you use recognition technology or manually type the data, you must identify the characteristics and processing details of the data on your documents. With this information, you can determine the data recognition requirements and other aspects of handling the data, including validations, lookups, verification, indexing, and data entry.

Identify the requirements for captured data:

- ▶ What fields should be manually entered at the batch level (for example, Scan Date, Expected Number of Documents, or Expected Number of Pages)?
- ▶ What fields should be captured at the document level (for example, Invoice Number, Invoice Date, or Invoice Total)?
- ▶ What fields should be captured at the line item detail level (for example, Item ID, Item Quantity, or Item Price)?
- ▶ For each document type, is data primarily machine printed or hand printed?
- ▶ For hand printed documents, is the print constrained or unconstrained?
- ▶ Are there pages that do not have data that must be recognized, such as attachment pages? It is common for forms to have instruction pages that are scanned but that do not have data on them.
- ▶ How is data located on the pages where you need to use recognition to read the data?
 - Fixed form layout. Fields are on specific zones where the location can be used to find the data.
 - Variable form layout. Fields have text labels where a search for the text label can locate the field.
 - Data is contained in a barcode.
- ▶ Is data validated by using an external database?
- ▶ What are the business rules for validating the values of the fields?
- ▶ Do fields have lists of valid values?
- ▶ Is it data optional or required?
- ▶ Does the data printed on the page conform to a repeatable pattern? (For example, Credit Memo Number starts with the letters CR followed by six numerics, a hyphen, and three numerics.)

5.3.5 Verification requirements

Verification intersects users with the documents. You must understand where these users are located and what tasks they are authorized to perform on each type of document. Business rules need to be applied that might mirror existing practices for handling paper-based data entry. Verification might also be desired as a quality control step to ensure that every image is readable.

Identify the requirements for verification:

- ▶ Will verification be handled in a central location or from remote locations?
- ▶ Are there business rules or policies that will require multiple verification steps?
- ▶ Who will perform verification?
- ▶ Does verification need to restrict access to specific document types by different groups of users?
- ▶ Do we need to display every document or page or can we display only documents or pages where we have exceptions?
- ▶ Will some documents require manual page identification by an operator?
- ▶ Based on the information gathered on input documents, captured data, and export requirements, how should low confidence data, invalid data, unidentified documents, and incorrectly identified pages be handled?
- ▶ When recognition results are high confidence, do you want an operator to view the document anyway?
- ▶ Do operators need to visit all fields with low confidence characters?
- ▶ Under which circumstances can the operator split out a document from the batch to finish processing the other valid documents in the batch? How should the split-out documents be handled?
- ▶ Should operators be able to mark document for deletion (documents will not be exported)?
- ▶ Should deletion trigger a follow-up process or automatic notification?

5.3.6 Export requirements

Identify the format, content, and target system or systems of the data and images for export:

- ▶ What is the document format for the exported documents for each type of document: TIFF, PDF, PDF with text, PDF/A, original input format, or other?
- ▶ Is the original image or the enhanced image used for export?
- ▶ Are color and gray scale images to be exported as color and gray scale?
- ▶ Is a specific file naming convention needed for the exported document?
- ▶ What are the document properties of the exported documents?
- ▶ Do the images have areas that need to be redacted?
- ▶ What are the target application systems for the exported data?
- ▶ What are the interfaces that are available in the target systems for ingesting the data?
- ▶ What data fields are exported to target application systems?
- ▶ Does the data need to be reformatted to accommodate the needs of the target application?

5.3.7 Volume and timing requirements

You must size and appropriately install and configure various Taskmaster components (remote/local clients, background processing, Fingerprint Service, Fingerprint Maintenance Tool). To assist with this task, create a matrix based on the following information:

- ▶ Sources of input
- ▶ Input volumes for each source
- ▶ Approximate number of unique documents (number of fingerprints)
- ▶ Peak periods
- ▶ Timing (processing windows) requirements

Identify the volume and time requirements:

- ▶ What are the input sources: scanner, fax, email, or other systems?
- ▶ How many document or image files are processed per day from each source?
- ▶ How many documents per document type are processed per day?
- ▶ For highly variable documents, such as invoices, how many different document formats are processed?

- ▶ What is the peak volume of documents and image files? Are there peak processing cycles daily, weekly, monthly, or annually?
- ▶ Are there peak volume requirements per day, or are there specific service level agreements about how quickly documents will be processed?
- ▶ Do existing paper files need to be scanned (backfile)? Quantify the volume and time frame for digitizing. Are the processing requirements different for historical documents compared to new documents?
- ▶ Is the ability to prioritize batches or to change the sequence in which the batches are processed required?
- ▶ What are the availability requirements for the system?

5.3.8 Administration requirements

Identify the administration-related requirements:

- ▶ What are the production reporting requirements? Compare Taskmaster standard reports to determine if custom reports formats are needed.
- ▶ What information needs to display for job monitoring? Compare Taskmaster monitoring views to determine if additional fields are required.
- ▶ What is the organization model for administering the system?
- ▶ What are the security requirements for authentication?
- ▶ How does the organizational security model map to the security requirements for the documents and capture functions?

5.4 Designing the capture for the auto claims scenario

In this section, we apply the design elements to the scenario that were introduced in Chapter 4, “Solution example” on page 123. We produce a high-level design that we implement in the following chapters. We describe the document hierarchy and the capture processing tasks that we implement.

5.4.1 Document hierarchy

The design document lists the expected document types and page type that we process in our application. We need to note any special document structure requirements. For our application, we considered three potential document types: claim documents, estimate, and invoice documents.

Claim document

A claim document is a form that is generated based on a conversation with the customer over the phone or through an online form. It documents information about the customer and the incident. The customer must sign the form and return it to the main office, by fax, scan, or email.

The format of the claim document is fixed. We control the layout of the document and have designed a single page for the customer to sign the form and date the signature date. We print a barcode on the claim page so that we can recognize the type of page and record the claim number.

Claim pages and claim attachment page

The claim document contains a claim page. It is the first page of a claim document, and each document has only one claim page.

Although we produce only a single page claim, we must be prepared to handle more pages. Customers often attach additional pages with additional explanations to the claim form. To accommodate this situation, we added a second page type, a claim attachment page. When we process a claim document, we expect only one claim page or a claim page with one or more attachment pages. The claim page is the first page of the document.

Estimate and invoice document

When we examined the estimate and invoice documents, we discovered that these two documents were identical. They were both produced by the same system. The only significant difference is that the estimate document has a title indicating that it is an estimate and the invoice document has a title indicating that it is an invoice.

As a result, we decide to define two types of documents within the Taskmaster system: claim and invoice-estimate. When we export, we store estimates and invoices in separate document classes in FileNet Content Manager.

Estimates

When a repair shop evaluates the damage to a vehicle, it prepares an estimate of the costs of the repair. The repair shop creates this document, in which the format is subject to change without notice. We have no control over the layout or contents. When a repair shop prepares an estimate, it is printed from their system in the format that they choose, which can be a multipage document. However, it will only contain one estimate per document.

We can also have our own adjusters produce repair estimates. However, for this scenario, we want a more difficult situation where the document format is beyond our control so that we can show how to handle the highly variable format.

Invoices

After a repair shop completes the repairs to a vehicle, it prepares an invoice that details the actual cost of the repair. Similar to the estimate, the repair shop generates this document, in which the format is subject to change without notice. We have no control over the form layout or contents. This document can be a multi-page document. However, it will only contain one invoice per document.

Estimate and invoice pages: Main page and trailing pages

The estimate and invoice documents have two types of pages: main page and trailing page. The *main page* is the first page, and only one main page exists. Each estimate can have many trailing pages. All of the pages have data that we can extract if we want to extract the detail line items. In this example, we extract the header data.

Fields

The data fields are designated for each page. The data fields are objects that are attached to a page. We can have a particular data field on several pages, but a particular data field can only be used once per page. When you attach rules to a field level object, and then use this field on an additional page, it immediately inherits all pre-existing rules.

Table 5-2 lists the data fields that are expected on the claim pages.

Table 5-2 Claim page fields

Field	Taskmaster field name	Description	Page source	Populated from
Incident Number	IncidentNumber	A number assigned by our company to identify this specific damage claim	We encode this number in a barcode and print it on the page.	Barcode
Policy Number	PolicyNumber	The insurance policy of the customer or contract number	We preprint this information on the page.	Database lookup
Policy Effective Date	PolicyEffectiveDate	The date when the insurance coverage begins for this policy	We preprint this information on the page.	Database lookup
Policy Expiration Date	PolicyExpiryDate	The date when the insurance coverage ends for this policy	We preprint this information on the page.	Database lookup

Field	Taskmaster field name	Description	Page source	Populated from
Insured Name	InsuredName	The name of the person who has insurance coverage from this policy	We preprint this information on the page.	Database lookup
Chassis Number VIN	ChasisNumber_VIN	A number that uniquely identifies the vehicle and is assigned by the vehicle manufacturer	We preprint this information on the page.	Database lookup
Injury	Injury	Indicates that an injury was associated with the incident	If an injury has occurred, the insured selects the appropriate check box on the form.	OMR
Vehicle Make	Veh_Make	The make of the vehicle (manufacturer)	We preprint this information on the page.	Database lookup
Vehicle Model	Veh_Model	The model name of the vehicle	We preprint this information on the page.	Database lookup
Vehicle Color	Veh_Color	The color of the vehicle	We preprint this information on the page.	Database lookup
Insured Signature	InsuredSignature	The handwritten signature from the insured	The insured writes their signature on the page.	OMR
Signed Date	SignedDate	The handwritten date from the insured	The insured writes the date on the page.	ICR

Table 5-3 lists the fields that are expected on the estimate and invoice pages.

Table 5-3 Estimate and invoice page fields

Field	Taskmaster field name	Description	Page source	Populated from
Document Title	Doc_Title	Estimate or Invoice	Printed on the page	OCR page text, search using regular expression to determine Estimate or Invoice
Policy Number	Pol_Number	The customer's insurance policy or contract number.	Not printed on the page	Database lookup

Field	Taskmaster field name	Description	Page source	Populated from
Claim Number	Claim_Number	A number assigned by our company to identify this specific damage claim.	Can be printed on the page	OCR or typed if not printed on the page
Vendor Number	Ven_Number	An identification number for the Repair Shop from our company vendor table.	Not printed on the page.	Database lookup and then matched from the Fingerprint ID
Reference ID	Ref_ID	The estimate or invoice number from the repair shop	Printed on the page	OCR
Reference Date	Ref_Date	The estimate or invoice date	Printed on the page	OCR
Reference Total	Ref_Total	The total amount of the estimate or invoice	Printed on the page	OCR

5.4.2 Capture processing tasks

The application has the following capture processing tasks:

1. Document acquisition

For our project, we implement centralized fax import, scanning, and distributed web-based scanning. Claim forms are mailed or faxed to our company. Faxes are imported from a central file system location, which is in the mailroom. Agents scan by using a web-based scanner.

Email capture was out of scope for our project but might be implemented by using the capabilities of the product.

2. Image enhancement

Images are enhanced by using the image enhancement features of Taskmaster that we are using to deskew the images and drop out horizontal and vertical lines.

3. Page identification

Claim pages are identified with a barcode printed on the form that contains a form identification code.

Claim attachment pages are identified by using rules. A claim attachment page is any page that follows a claim page.

Estimate and invoice pages are identified with a separator page containing a barcode. Alternatively, they are received to a specific fax number that is dedicated to receiving invoices and estimates from vendors.

4. Document assembly

Documents are assembled by using document integrity rules that are expressed in the document hierarchy.

5. Recognition

Claim data is recognized by using barcode recognition: OCR/S to read machine print text and ICR/C to read handwriting.

Expense and invoice data is recognized by using OCR/S.

6. Fingerprinting

We use fingerprints to differentiate between multiple formats of the same page type.

The Intellocate actions are used to automatically make new fingerprints if needed. In our use case, we described the estimates and invoices that came in from the repair shop. Each time we see a new layout for an estimate or invoice, we will capture it and make a new fingerprint.

7. Locating data

Claim data is located by using zones. Expense and invoice data is located by using Intellocate.

8. Validation

We do database lookups to an IBM DB2 database to validate the incident number and policy data, from the fields on the claim page.

We validate the vendor information with database lookups to a DB2 database.

We set a threshold for recognition confidence so that low confidence characters and fields are flagged.

9. Routing

Exceptions are routed to a verify operator. Items that require follow-up by customer service or agent are routed by using the business process workflow for resolution.

10. Verification

Verification occurs at a central location. Low confidence results or validation errors are presented to operators.

11. Export

For our use case, we use a combination of PDF/A and TIFF file export. The claims are stored in TIFF, while the estimates and invoices are stored as PDF/A documents. We revert to the original image because it was scanned before enhancement.

We export the documents to the FileNet Content Manager. We include corresponding document classes that are defined with document properties for the claim, estimate, and invoice documents where they are stored and processed by the business.

Extracted data is exported to an XML file that is stored in a designated folder on our file server.

To implement the solution, follow the step-by-step instructions in the next chapters:

- ▶ Chapter 6, “Implementing the capture solution” on page 179
- ▶ Chapter 7, “Adding a document type to an existing application” on page 253
- ▶ Chapter 8, “Implementing the business process component” on page 269



Implementing the capture solution

This chapter explains how to implement the capture solution by using the solution example as outlined in Chapter 4, “Solution example” on page 123. To configure the system, you use the IBM Datacap Taskmaster Capture (Taskmaster) development tool Datacap Studio. You see how to design a capture workflow and create an application. The chapter also explains how to create actions, rules, and rule sets, how to configure document hierarchy and task profiles, and how to set up the verification panel.

This chapter includes the following sections:

- ▶ Configuring the Datacap application
- ▶ Zones and fingerprints
- ▶ Task profiles overview

Before you read this chapter, you must have already installed the Taskmaster software. If you have not installed it, follow the installation information in Chapter 13, “Installation, migration, and application reuse” on page 453.

6.1 Configuring the Datacap application

This section guides you through the creation of a Taskmaster application by using the solution example of the insurance company introduced in Chapter 4, “Solution example” on page 123. It shows you how to create the claim form-related portion of the application. By following the solution example, you can see many of the common functions that are used by the Datacap applications.

Creating the claim form portion of the application entails the following tasks:

1. Installing and configuring the scanner on the Datacap client.
2. Creating the required rule sets.
3. Adding the rule sets to the task profiles.
4. Adding the rule sets to the document hierarchy.
5. Creating the Verify user interface in Batch Pilot.

6.1.1 Creating a blank application

To create an application from scratch using Datacap Studio and Batch Pilot, complete these steps:

1. Launch Datacap Studio by selecting **Start** → **Programs** → **Datacap** → **Datacap Studio**.
2. Click **Close** when you are prompted for the application you want to open.

Datacap Studio opens with a blank view as shown in Figure 6-1.

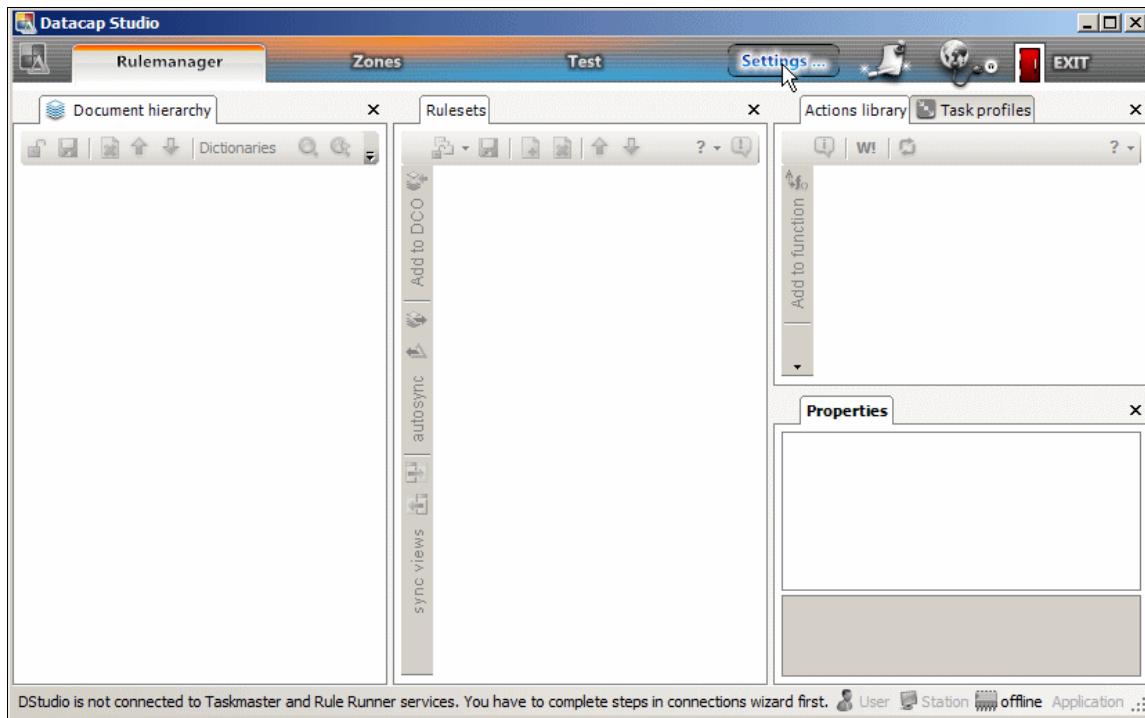


Figure 6-1 Blank view of Datacap Studio

3. Click the **RRS** tab, and configure the web service and Rulerunner general settings (Figure 6-2):
 - a. In the Service box, configure the service. We select **Work offline**.
 - b. In the Rulerunner general settings box, Specify the path to the global rule sets and actions location. This path usually points to a directory that is updated during product update on the project server. Do not store customized rule sets and actions here. We enter the c:\datacap\RRS\ path. Then we select **Save older version of rules in archive(s)**.

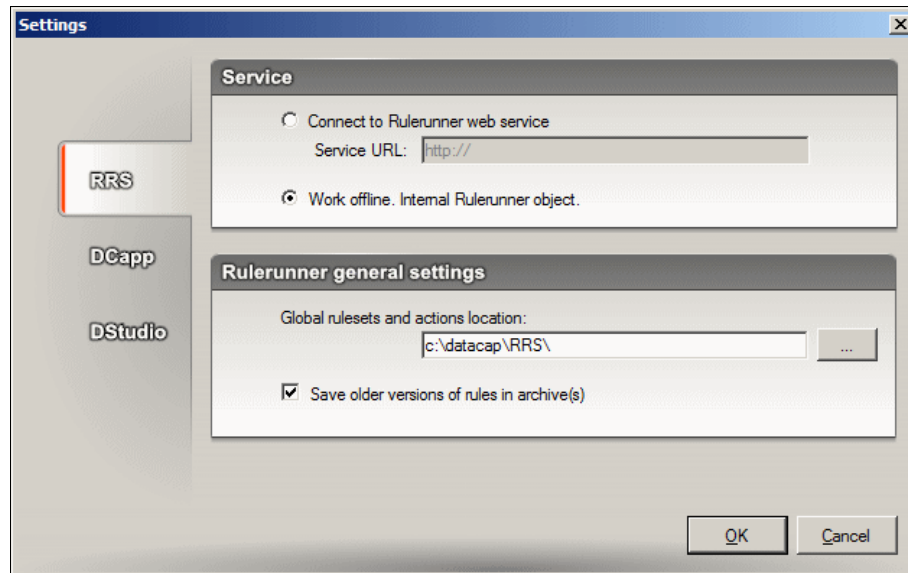


Figure 6-2 RRS settings

4. Click the **DCapp** tab, and enter the path that holds the main application management files for the client (Figure 6-3). This path usually points to the file in the local Datacap directory. To improve administration, alternatively you can place it on a network drive. For this case study, we use the default value. We also select **Save older versions of application in archive(s)**.

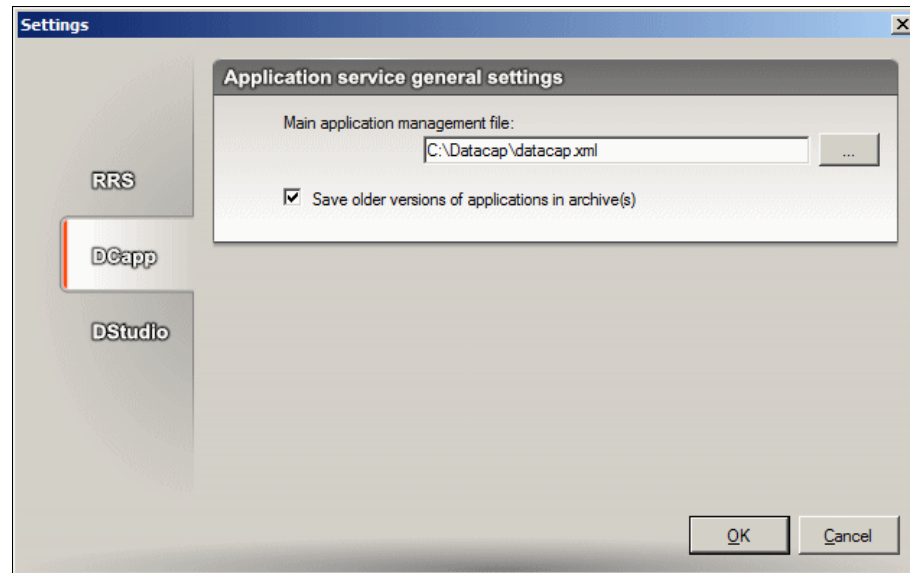


Figure 6-3 DCapp setting

5. Click the **DStudio** tab (Figure 6-4). Optionally, select the **Transparency** option, which defines the transparency of the pop-up windows within Datacap Studio. For this use case, we do not select the check box, which is the default.

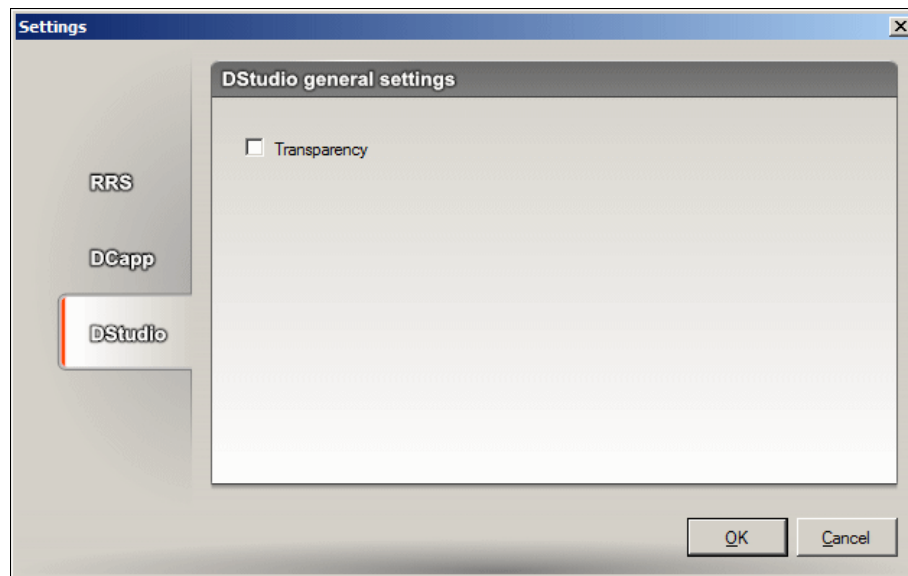


Figure 6-4 DStudio transparency setting

6. In the Settings window, click **OK**.
7. Click the **Application Wizard** icon next to “Settings” at the top of the Datacap Studio window (Figure 6-5) to open the application wizard to start creating an application.

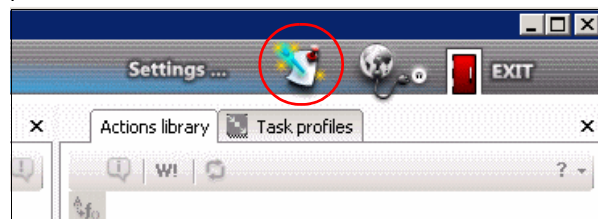
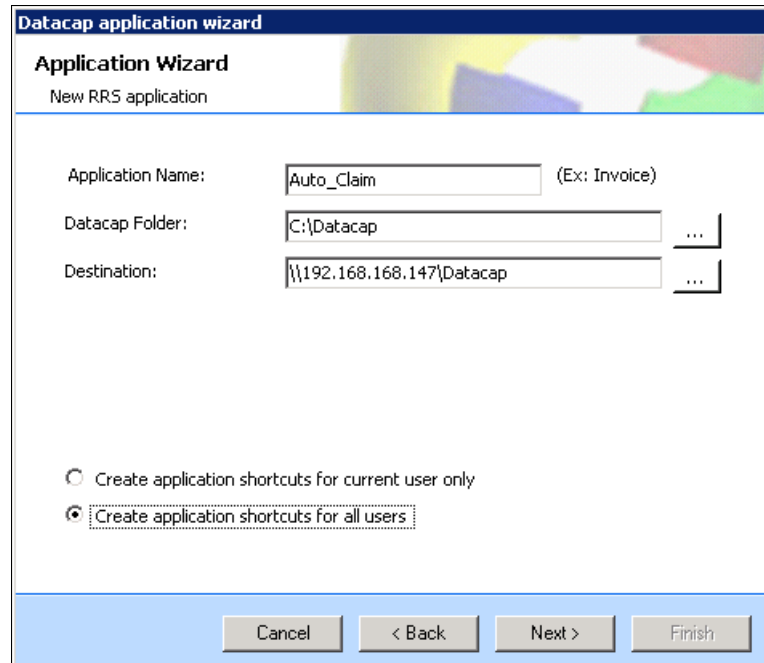


Figure 6-5 Clicking the Application Wizard icon

8. Select **Create a new RRS application**.

9. In the Datacap application wizard—Application Wizard window (Figure 6-6), complete these steps:
 - c. Enter the name of the new application. Users must enter this name in the client when they log on to an application.
 - d. Enter the Datacap Folder, which points to the executables of the clients. You can set an alternative path on your local machine if you do not follow the standard installation.
 - e. For the Destination field, which points to the project directory, enter a network path to make the project available to others. You can use Universal Naming Convention (UNC) notation for the server.
 - f. Optional: Select the option to create an application shortcut for either the current user only or for all users.
 - g. Click **Next**.



The screenshot shows the 'Datacap application wizard' window, specifically the 'Application Wizard' step for a 'New RRS application'. The window has a title bar with the text 'Datacap application wizard'. Below the title bar, the text 'Application Wizard' and 'New RRS application' is displayed. The main area contains three input fields: 'Application Name:' with the value 'Auto_Claim' and a hint '(Ex: Invoice)', 'Datacap Folder:' with the value 'C:\Datacap', and 'Destination:' with the value '\\192.168.168.147\Datacap'. Each of the last two fields has a browse button (three dots) to its right. At the bottom, there are two radio buttons: 'Create application shortcuts for current user only' (unselected) and 'Create application shortcuts for all users' (selected). The bottom of the window features a blue bar with four buttons: 'Cancel', '< Back', 'Next >', and 'Finish'.

Figure 6-6 Specifying the application name and paths

10. In the Document Hierarchy, Fingerprints, and Add sample images windows, click **Next** to accept the default settings. The options are handled later.
11. Click **Finish** to start creating of the new application.
12. When the wizard is finished, review the summary information of the new application (Figure 6-7), and then click **Close** to close the wizard.

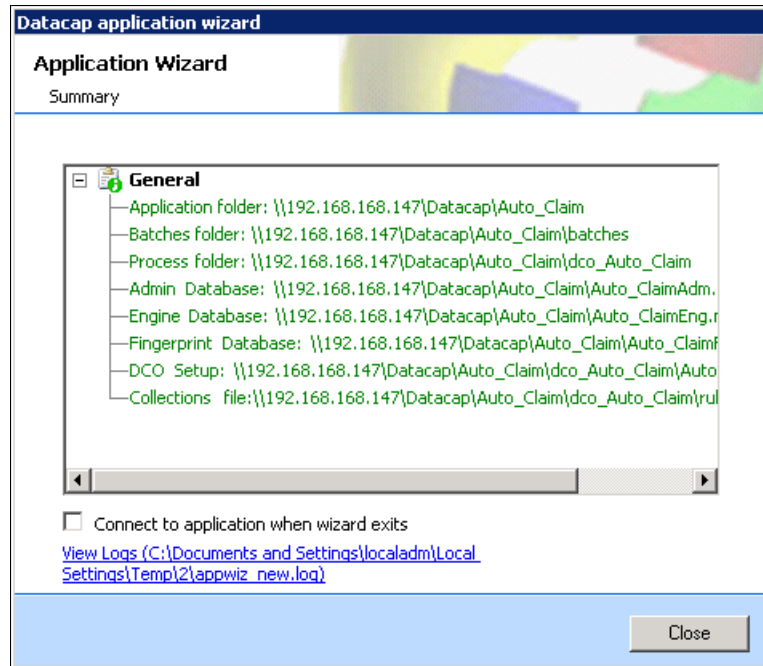


Figure 6-7 Application wizard, summary

The wizard creates the directories with the required files in it. The wizard also registers the new application in the Taskmaster environment. On a 32-bit Windows system, the wizard creates the shortcuts in the Windows Start menu.

Changing the Taskmaster Server location

Taskmaster Application Manager manages the configuration of applications. For our use case, we must correct the location of the Taskmaster Server.

To change the location of the Taskmaster Server, complete these steps:

1. Select **Start** → **Programs** → **Datacap** → **Taskmaster Client** → **Taskmaster Application Manager**.

2. In the left side of the Taskmaster Application Manager window (Figure 6-8), select the new application **Auto_Claim**.

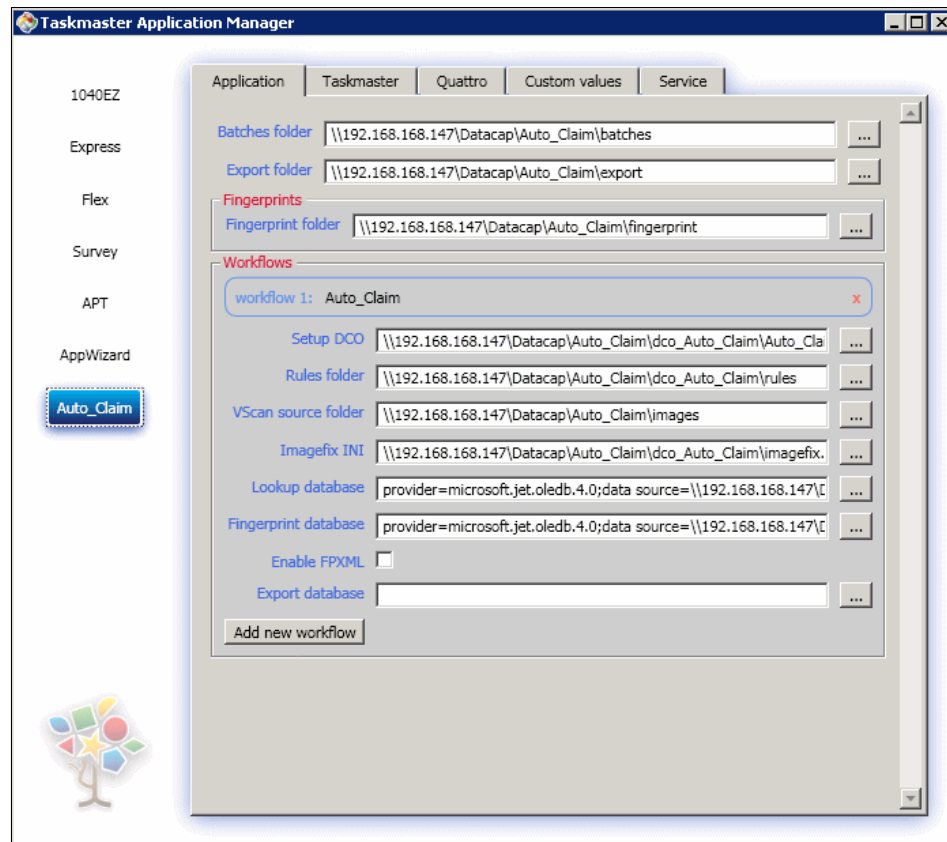


Figure 6-8 Taskmaster Application Manager window

3. Click the **Taskmaster** tab, and change the server name and the address entry appropriately. You can enter several server addresses. If the first server is unreachable, the Taskmaster clients automatically try the next server.

4. Go back to the Datacap Studio application. Click the **Connection wizard** icon (the second button from the right side) in the upper right corner of the window (Figure 6-9) to start the connection wizard and connect to the new application.

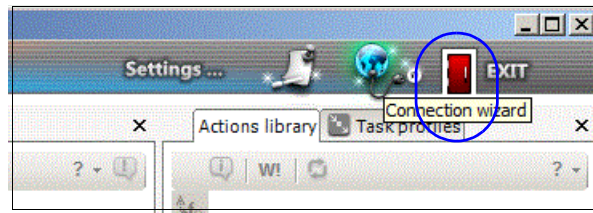


Figure 6-9 Connection wizard

5. Select the new application **Auto_Claim** and click **Next**.
6. In the Taskmaster Login window (Figure 6-10), enter the user ID and the password for station 1. Then click **Finish**.

A screenshot of the 'Taskmaster Login' window. The title is 'Taskmaster Login'. Below the title, it says 'Please provide Taskmaster authentication details which you want to use to login to Taskmaster below.' The window contains the following information:
Taskmaster Server
Morpheus.datacap.com
Administrator database MS Access
\\192.168.168.147\Datacap\Auto_Claim\Auto_ClaimAdm.mdb
Engine database MS Access
\\192.168.168.147\Datacap\Auto_Claim\Auto_ClaimEng.mdb
Below this information is a login form with the following fields:
☐ NT authentication
User ID: admin
Password: xxxxxx
Station ID: 1
At the bottom of the form, there is a checkbox labeled 'Hide this step if successfully performed on Datacap Studio startup'. Below the checkbox are four buttons: '< Back', 'Next >', 'Finish', and 'Close'.

Figure 6-10 Logging in to the Connection wizard

6.1.2 Setting up the document, pages, and fields

After creating a new blank application, you must set up the document, add or update the associated pages to the document, and add the associated fields for each page.

For our use case, we name our document `Auto_Claim`. The document contains two pages, `Claim_Pg` and `Claim_Attachment_Pg`. The first page, `Claim_Pg`, also contains 25 fields.

To set up the document, pages, and fields, complete these steps:

1. Start the editing mode:
 - a. At the top of the window, click the **Rulemanager** tab.
 - b. On the **Document hierarchy** tab, click the **Lock DCO for Editing** icon. For information about the structure of the Document Hierarchy (DCO), see 5.2.1, “Document hierarchy” on page 142. Now you are ready to add or update content to the project.
2. Rename the document (Figure 6-11):
 - a. Click the **Document** DCO to select it.
 - b. After a pause, click the **Document** DCO a second time, and rename it to `Claim_Doc`.

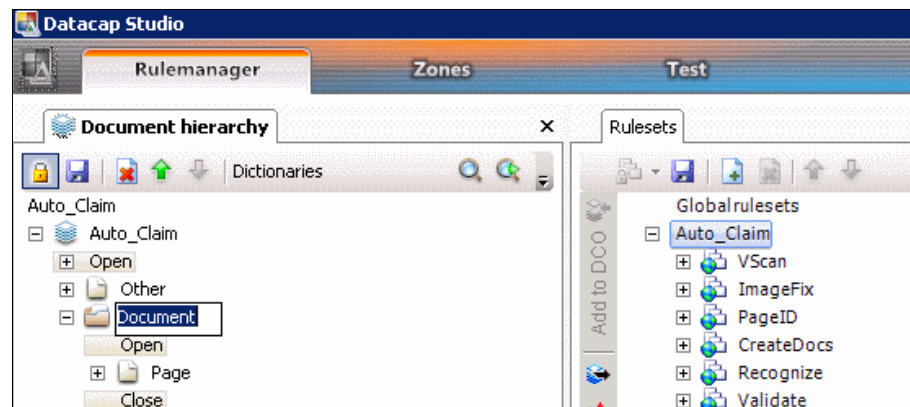


Figure 6-11 Creating a DCO and renaming it

3. Repeat step 2 for the page, and then rename it to `Claim_Pg`.

4. Add another page to that document:
 - a. Go to the document, right-click it, and select **Add Page**.
 - b. Rename the second page as needed. For our use case, we name it Claim_Attachment_Pg as supplementary pages.
5. Add fields to your page (Figure 6-12):
 - a. Right-click the page (**Claim_Pg** in this example), and select **Add multiple** → **Fields**.
 - b. Enter the number of fields you want to add. In this example, we enter 25.

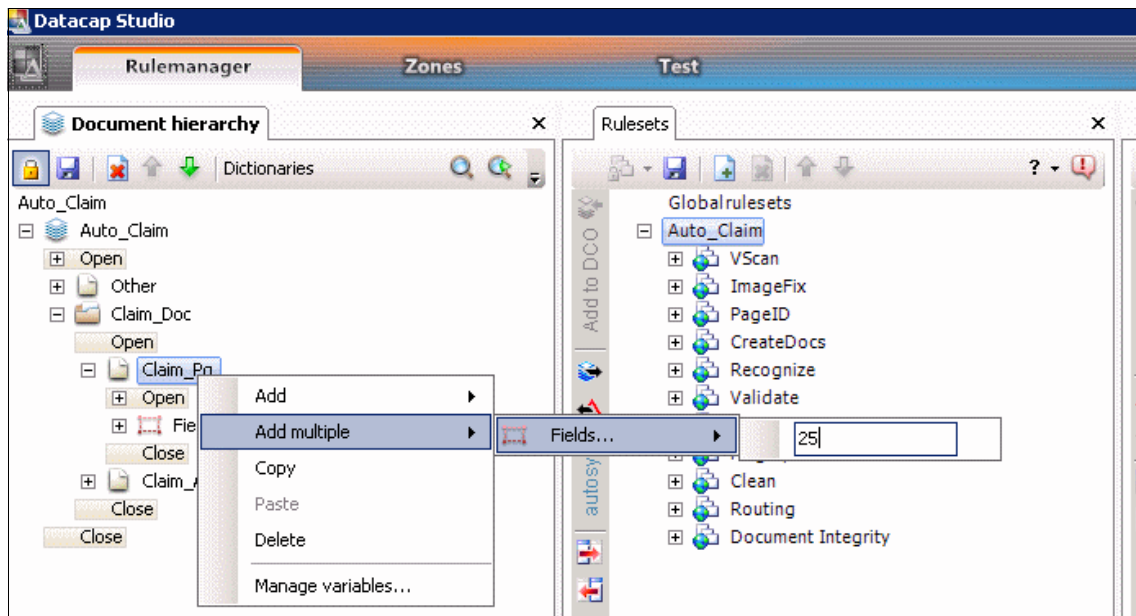


Figure 6-12 Adding multiple fields in Datacap Studio

6. Save and unlock the Document Hierarchy.

6.1.3 Setting up the physical scan device

Prerequisite: The scanner must be installed and configured with the Image and Scanner Interface Specification (ISIS) driver at the Windows level. For more information, see the documentation for the scanner.

To set up the IScan.bpp file, complete these steps:

1. In the DCO_ directory, look for the iScan.bpp file. If you cannot find this file, copy the kScan.bpp file as the iScan.bpp file.
2. Launch Batch Pilot by selecting **Start** → **Programs** → **Datacap** → **Batch Pilot** and click the **Batch Pilot** executable.
3. In Batch Pilot, select **File** → **Open Project**.
4. Navigate to your DCO directory by using the fully qualified path, and select the **iScan.bpp** file.
5. If you see a warning message (Figure 6-13) indicating that the file is not found, click **OK**.

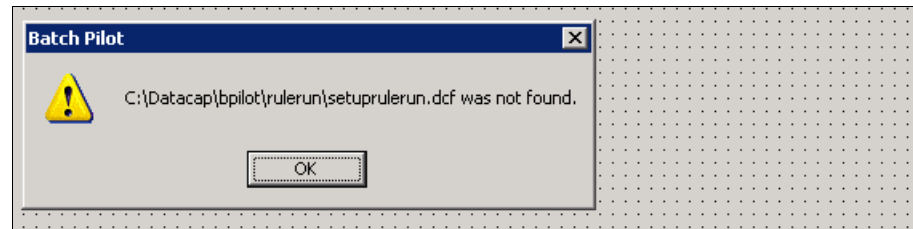


Figure 6-13 Batch Pilot, warning

6. In the Batch View window at the bottom of the Batch Pilot application, right-click the **Form Path** column of the **SetupForm** line and select **Pick form** (Figure 6-14).

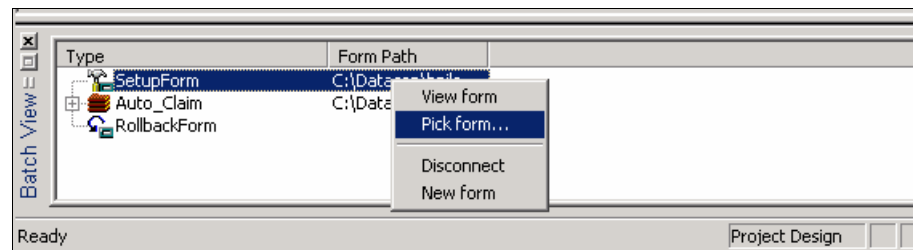


Figure 6-14 Selecting Pick form in Batch Pilot

7. Open the C:\Datacap\BPilot\ISScan\isscan.dcf file, which opens the corresponding form in the upper window.
8. Repeat steps 6 and 7 for the Auto_Claim entry the Form Path.
9. Select **File** → **Save Project** to save this project.
10. Select **File** → **Exit** to exit Batch Pilot.
11. If you are prompted to save the project, click **Yes**.

6.1.4 Creating a module in Taskmaster

Prerequisite: The scanner must be installed and configured with the ISIS driver at the Windows level. For more information, see the documentation for the scanner.

To create a module in Taskmaster, complete these steps:

1. Open the Taskmaster Client.
2. Open the **Auto_Claim** application.
3. Click the **gold key** icon in the middle of the upper icon bar (Figure 6-15) to open the Taskmaster Administrator.

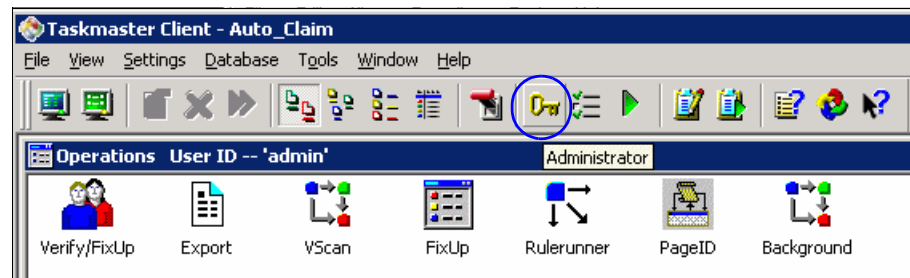


Figure 6-15 Starting Taskmaster Administrator in the Taskmaster Thick Client

4. Click the **Modules** tab, and then click **Add**.
5. In the right frame, enter a value for the module ID and its description. Change the type to **Batch creation**, and change the program name to **Batch Pilot**. The parameters point to the .bpp file that you just created.

6.1.5 Creating a job within the Taskmaster Client

We want to create a job. The New Application wizard creates three basic jobs (Main, Web Main, and Fixup). Because our use case requires multiple jobs, we use these jobs as patterns to create our new jobs.

To create a job within the Taskmaster Client, complete these steps:

1. Click the **Workflow** tab.
2. In the left pane, select **Main Job**, and then click **Copy**.
3. In the Copy Jobs dialog box (Figure 6-16), enter a name for the new job. For our use case, we enter **Claim Thick**. Click **OK**.

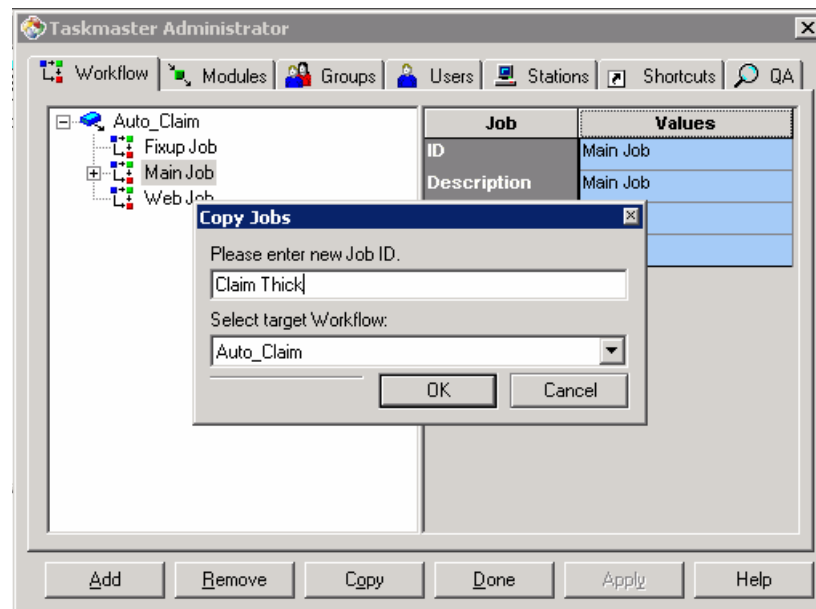


Figure 6-16 Copy Jobs dialog box

A copy of the source job is created, except for the name that you assigned it.

4. Remove any unnecessary tasks from the job.

The New Application wizard places the VScan task by default. We need to remove it from our job:

- a. Click the new job, **Claim Thick**, to open its subcategories.
- b. Right-click the **VScan** task, and then select **Remove** to remove the VScan task.

5. Add necessary tasks for the existing job.

For our use case, we need to add a task that control the thick client physical scanner:

- a. Right-click the **Claim Thick** job, and then select **New** → **Task** (Figure 6-17).

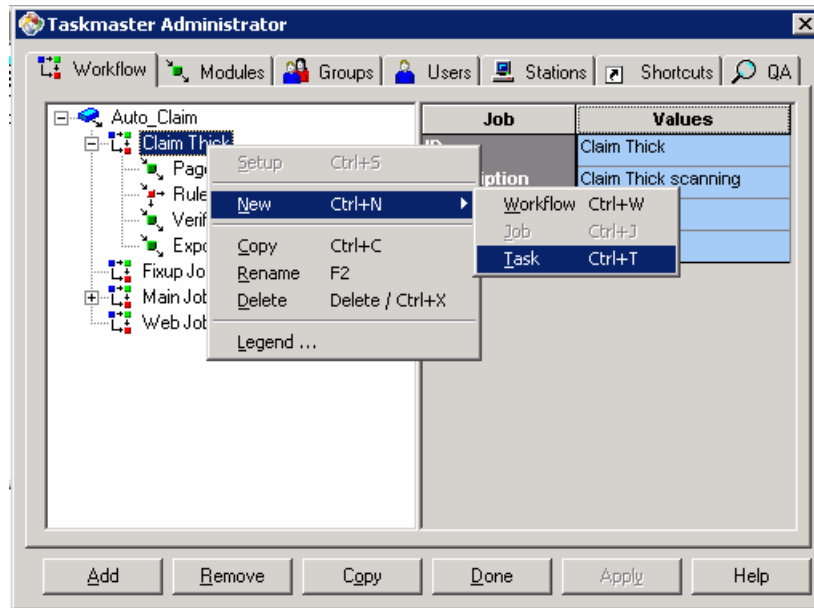


Figure 6-17 Adding a task

- b. Enter a name for the new task. We enter iScan.
- c. Enter a description for the new task.
6. Click the **Module** frame, and then select the **iScan** module from the list. By using the Ctrl and Arrow up keys, move this task to the top of the job. If you do not move up the task, the system displays an error message.
7. Click **Apply**.

Startbatch module: If a job contains the Startbatch module, the task that calls that module must be the first task in the job. You are limited to one Startbatch task for each job.

6.1.6 Setting up the iScan task

To set up the iScan task, complete these steps:

1. Click the **Setup** button in the right frame. The first time you click it, the system prompts an error message indicating an invalid procedure. Ignore this error message, and click **OK** and continue.

ISIS driver: The iScan task interacts with the ISIS driver. Actual controls might vary depending on the type of scanner you use.

2. Enter the path to the `settings.ini` file. This file stores the scanner settings for your particular scanner. Store this file in a location that is local to the scanner so that you can use the same iScan task for different ISIS scanners in your environment. This setup must be executed locally at every scanner in your environment.
3. Enter the path to the `ImprinterScript` file, and include the file name in the path. This file is used to interface with the imprint feature (if available) of the scanners so that you can control what is written on the paper by the scanner imprinter during the scan process.

For our use-case scenario, the scanner does not support the use of an endorser. Therefore, we leave this line blank.

4. Configure the StartBatch panel. In this optional panel, you enter additional information at scan time. With advanced development, you can enter a receive date, for example. The information that you enter in the StarBatch panel is available to the entire batch. For our use case, we do not use a StartBatch panel. Therefore, we delete the text from the StarBatch File: (with path) field (highlighted in Figure 6-18).

Figure 6-18 shows the Scan task setup that we entered for our scenario.

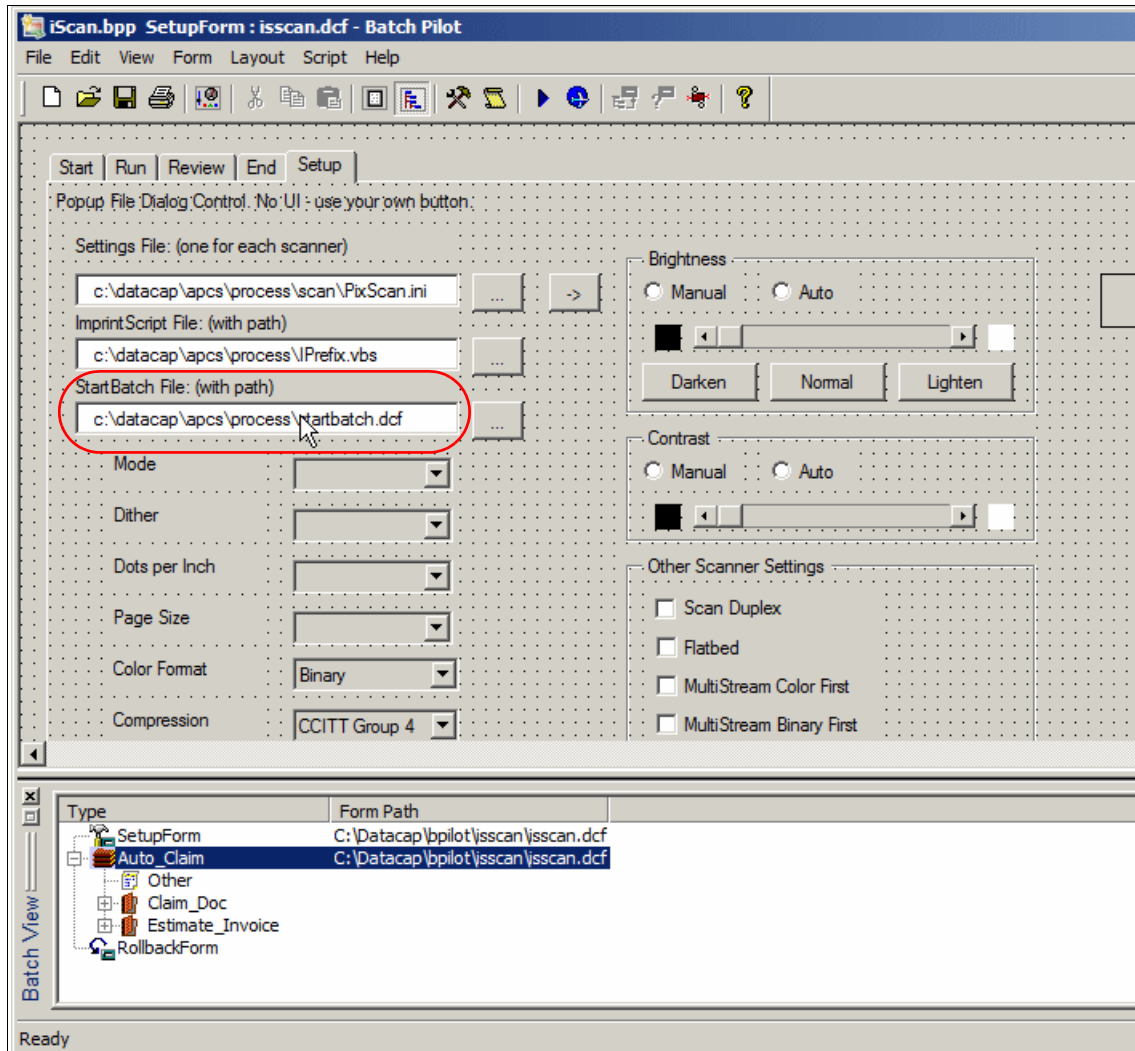


Figure 6-18 Setting up the Scan task

5. Select the appropriate scanner for your system:
 - a. Click **Select Scanner**. The iScan setup then scans your computer for the matching combination of the ISIS driver you have installed and the physical scanner that is connected (to your machine).
 - b. Select your specific scanner. For our case, we select **Epson S80**.
 - c. Enter additional information, depending on the scanner and the matching ISIS driver.
 - d. When you are finished setting up the scanner values, click **Done** to save all of the scanner settings that you selected in the .ini file that you specified earlier.

Figure 6-19 shows the result of the scanner setup for our use case. The settings are specific to the scanner that you select and the matching ISIS driver.

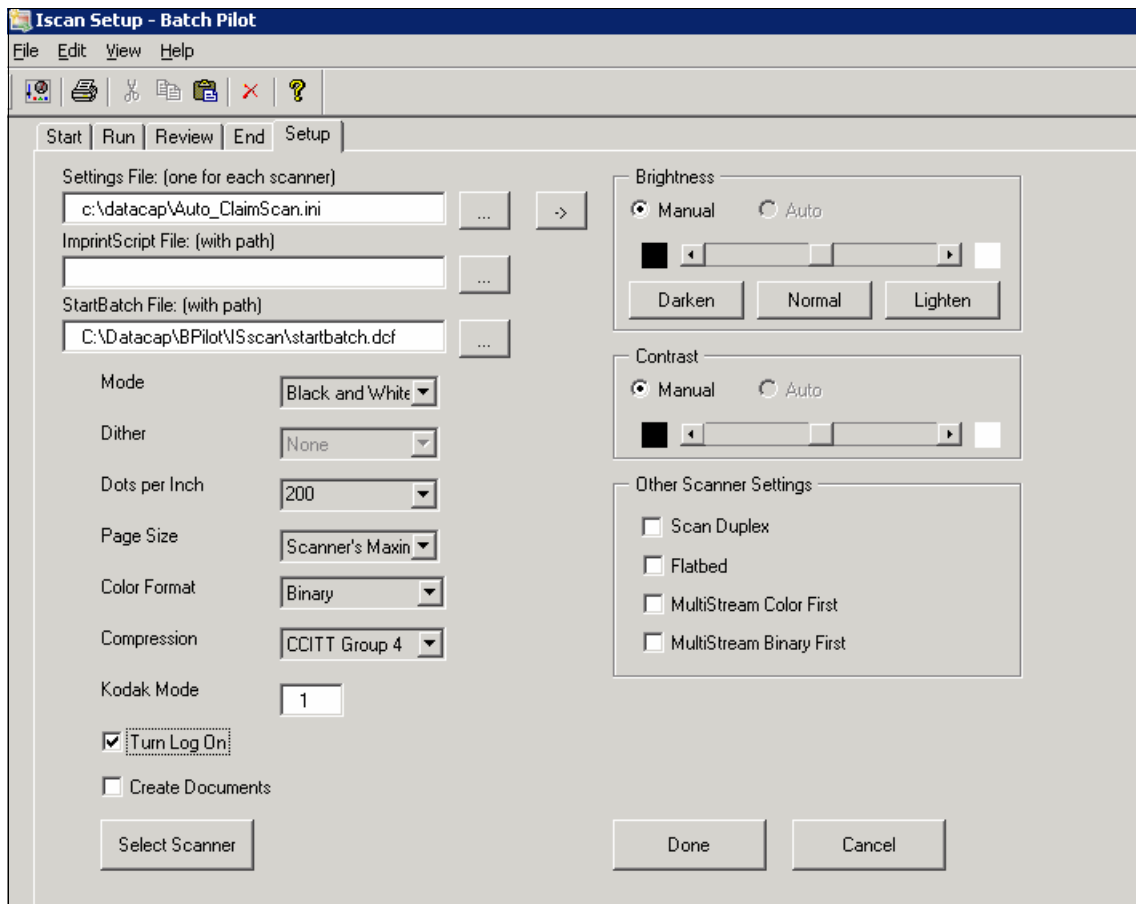


Figure 6-19 Configuring the scanner

For more information, see the document input section in *Taskmaster Application Development Guide*, SC19-3251.

Setting up the Scan icon

After you create the scan module, connect it to a task, and then set up the scanner behavior:

1. In the Taskmaster Administrator window, click the **Shortcuts** tab.
2. Click **Add**, which places the cursor to the input box for the Shortcut ID.
3. Enter in the name that you want for this shortcut. For our use case, we called this IScan. Enter a description for this short cut, and then
4. Click **Apply** to add the new shortcut you created to the Shortcuts list.
5. Define the settings for this shortcut. Select the shortcut in the left frame. Then in the right frame, select the Batch Selection Mode and the Permissions.

With the Batch Selection mode, you define, for the task, which batch to do next:

- The Auto mode automatically selects the next available batch. This batch is determined by using First In First Out (FIFO) technology. For a batch creation task, it automatically creates a batch.
- The Manual mode shows a list of all available batches, and the user selects which batch they want to run.
- The Manual for Hold mode only shows a list of all the batches that the current user has on hold. If that user has no batches on hold, the system reverts to the Auto mode.

When a user initially starts, the system prompts the user with the three choices. The user selects a choice. As long as the current session is active, that choice is always used.

We normally run the Scan task in Auto mode.

6.1.7 Setting up the PageID task

The PageID task uses Batch Pilot with a specific project file (a .bpp file). This file contains information such as the path to the DCO file, the path to the batches directory, a connection to a specific Task Profile entry, and several other values. The Task Profile connects the PageID task to a collection of specific rule sets to run, which is developed and maintained by Datacap Studio.

Setting up the PageID task profile

To set up the PageID task profile, complete these steps:

1. Open Datacap Studio.
2. Select the project.
3. Navigate to and click the **Task Profile** tab.
4. Move the PageID task in front of the ImageFix task.

On the **Task Profile** tab, tasks are listed in the order in which they run. The Application Wizard created ImageFix followed by the PageID tasks. Because we use barcodes for Page ID, we want to perform the PageID task before we do any image fix on the page. Therefore, we need to move the PageID task in front of the ImageFix task.

To move the task:

- a. Lock the task profile for editing.

On the **Task** tab, click the blue **padlock** icon in the upper left corner to lock the Task Profiles for editing. The blue icon then changes to a golden locked padlock.

- b. Expand the task profile.
- c. Select the task that you want to reorder, and then rearrange the task.

Click the **green** arrow keys to move the rule set up or down to rearrange. For the use case, we move PageID before ImageFix (Figure 6-20).

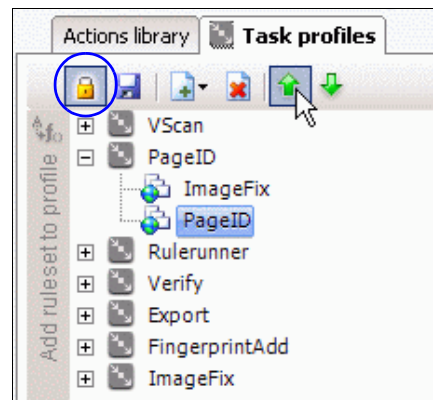


Figure 6-20 Rearranging rule sets

- d. Unlock the task profile for editing.

Click the gold **padlock** icon to unlock the task profiles. The task profile is now ready for operation.

Setting up the PageID rule set

The PageID rule set is used to automatically identify the pages that come in from the scanner. In our use case, for the claim application, we expect to receive two types of pages, Claim_Pg and Claim_Attachment_Pg. The claim page is identified by a barcode with a value of CLAIM-29A. If a page does not have this barcode value, it is defaulted to a page type of Claim_Attachment_Pg.

To set the rule set, we delete the existing function attached to the PageID rule set. The existing function is added when we create a new application. The PageID rule is attached to the page type “Other,” which is the default page type produced by the Scan task.

To make changes to the existing rule set, complete these steps:

1. Remove a function from an existing rule:
 - a. Select the rule set.
 - b. Click the **Lock/Unlock Ruleset** button.

Locked and unlocked items: DCOs and task profiles are locked and unlocked in this one step. Rule sets are locked and unlocked individually.

- c. Right-click the function you want to remove, and then select **Remove** (Figure 6-21).

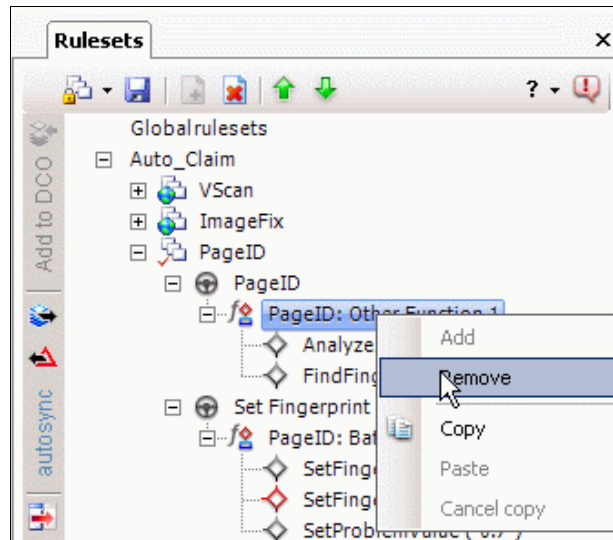


Figure 6-21 Remove function

2. Add a function to an existing rule. Right-click the rule set to which you want to add the function, and then select **Add Function** (Figure 6-22).

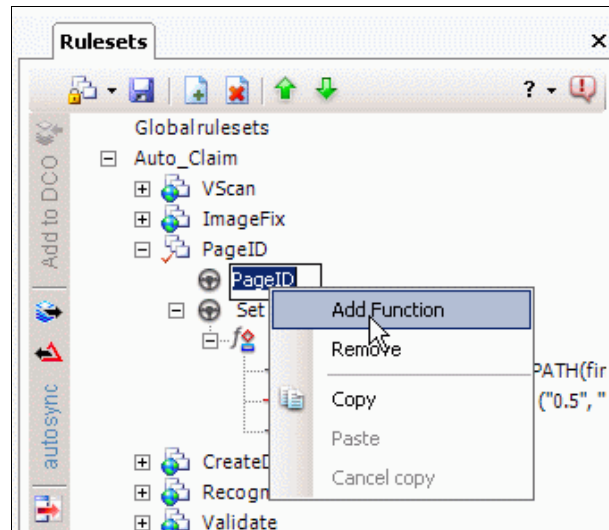


Figure 6-22 Adding a function

This step adds a function under the rule. The default name of the new function is "Function," with sequential numbering such as "Function1."

Although not necessary, consider renaming this function to a more descriptive name that indicates what it does. In our use case, we rename Function1 to Claim Id by Bar Code.

3. Repeat step 2 for each the following functions:
 - Page after Claim Page
 - Page after Claim Attachment Page

When you are done, you have three functions under the PageID rule:

- Claim Id by Bar code
- Page after Claim Page
- Page after Claim Attachment Page

4. Add an action to a function:
 - a. Click the function to which you want to add an action.
 - b. From the Actions Library, select the action that you want to add. For our use case, we select **MatchBarcodeBP**.
 - c. Click the **Add to Function** button.

5. Set the parameters for the action:
 - a. Under Function, click the **MatchBarcodeBP** action.
 - b. On the **Properties** tab, click Parameter, and then enter the value CLAIM-29A.
 - c. Press Enter to apply the parameter.

This action returns TRUE if the specified barcode value is found.

6. Add the SetDCOType("Claim_Pg") and Get2DCode() actions (Figure 6-23). In the lower center frame, you can choose to enter comments for the actions, functions, rules, and rule sets. You can add the comments to make your rule set more clear to other users.

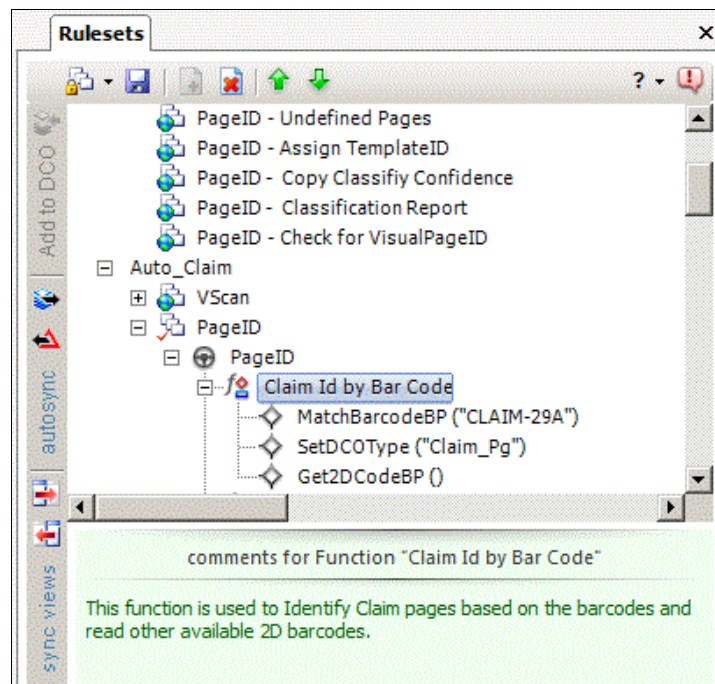


Figure 6-23 Setting PageID based on a barcode with comments

Case sensitivity: When using the SetDCOType() action, remember that you are calling a case-sensitive object. Ensure that you use proper capitalization.

Setting up the ImageFix rule set

The ImageFix rule set is used to clean up the image. The purpose of cleaning the image is to get the best recognition results possible from the document.

In this use case, we switched the order of PageID and ImageFix in the task profile because, when the ImageFix rule set runs, depending on the settings, it can damage a barcode. The ImageFix rule set normally contains two rules, one attached to the batch level and one attached to the Page object. For our use case, we use these two existing rules and add an additional action to the Enhance Image rule. We also change the object to which the Enhance Image rule is attached.

ImageFix load settings

The ImageFix rule set is attached to the batch level. It loads preconcerted settings on how you want image processing to behave. It uses the LoadSettings() action with the “@APPPATH(imagefix)” parameter. This parameter indicates to the action to use the application path, which is the path where your DCO directory resides. The ImageFix rule set directs it to use the imagefix.ini file in that application path.

Enhance Image rule set

The Enhance Image rule set, by default, is attached to the “Other” page object. We want to disconnect this rule set from the “Other” page object and attach it to the “Claim_Pg” page.

Figure 6-24 shows the setup.

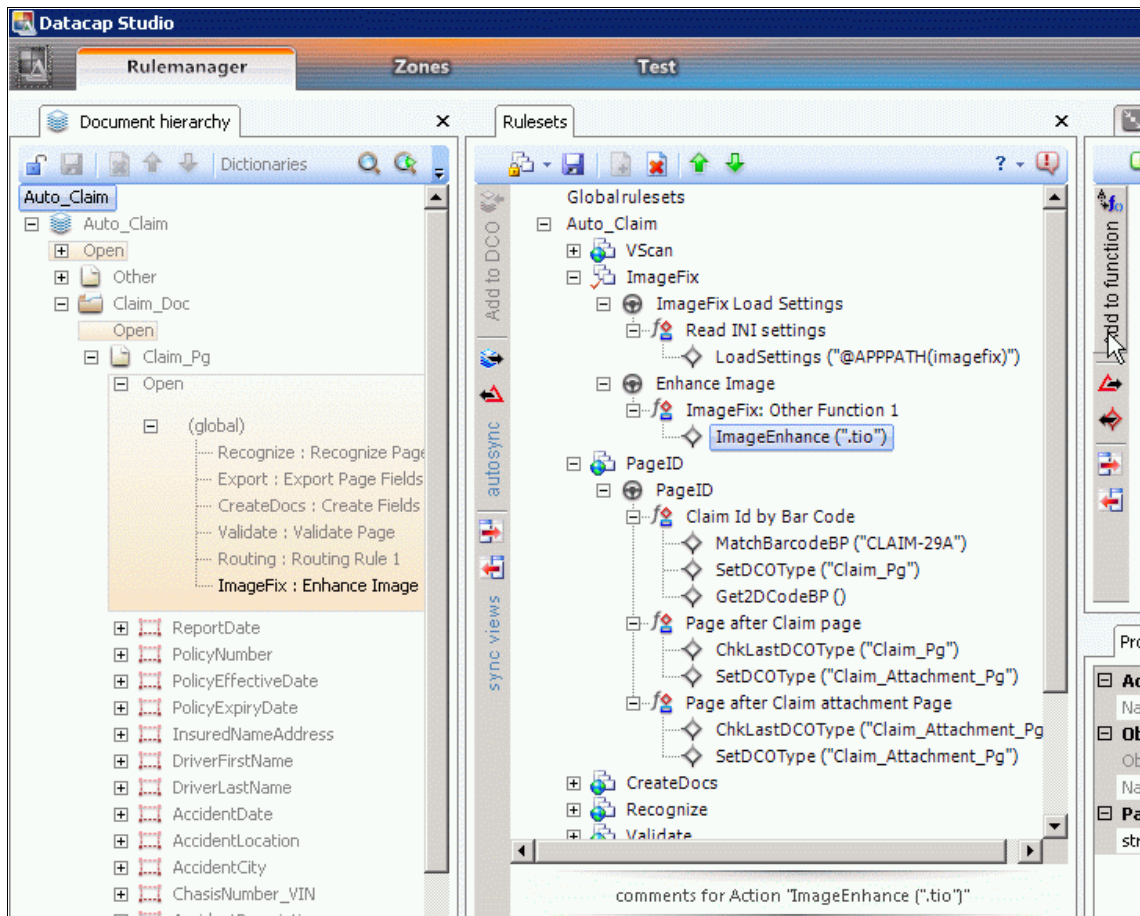


Figure 6-24 Attaching a rule to a DCO

To remove a rule from an object, complete these steps:

1. On the **Document hierarchy** tab, click the **padlock** icon to lock the DCO for editing.
2. Expand the **“Other”** page object to expose “Open”/(global).
3. Right-click **ImageFix: Enhance Image**, and select **Delete**.

To add the rule to a new object, complete these steps:

1. Click the **Claim_Doc** to expose the Page Object "Claim_Pg."
2. Click the rule you want to add.
3. Click the **Add to DCO** button.
4. Under Document Hierarchy, click **Save**.
5. Click the **padlock** icon to unlock the DCO for editing, allowing it to be used.

Testing: Now test your application through the PageID task.

Figure 6-25 shows the test result of the pageID.xml file.

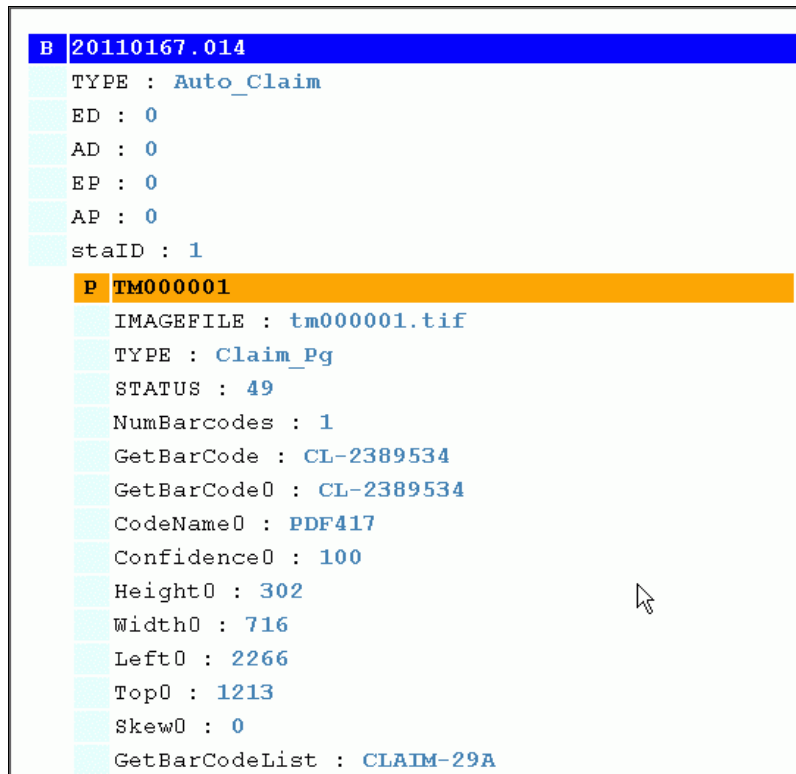


Figure 6-25 Test result of the Page ID file

6.1.8 Setting up the Rulerunner task

The Rulerunner task is an unattended task that runs after the Page ID task and before the Verify task. The purpose of this task is to perform background processes. What happens here varies from project to project, although normally it includes basic elements. The Rulerunner task normally runs on a system in the network room. It does not have a user logged in. Although it runs on a desktop computer, running it on a server is preferred because of the heavy traffic that goes through it. In the Datacap portion, this system performs the most work.

For our use case, we use the rrsRuleRunner module (Figure 6-26).

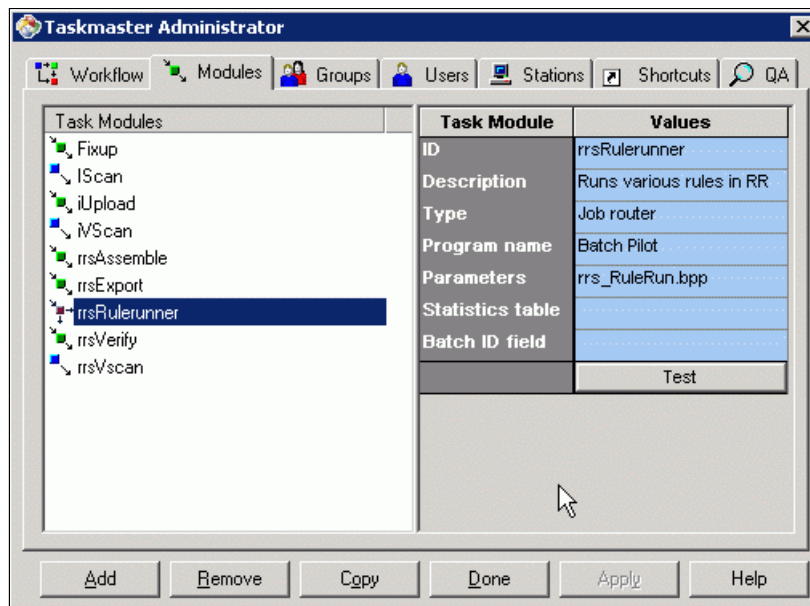


Figure 6-26 Setup of the rrsRuleRunner module

This module calls the Rulerunner task profile (Figure 6-27).

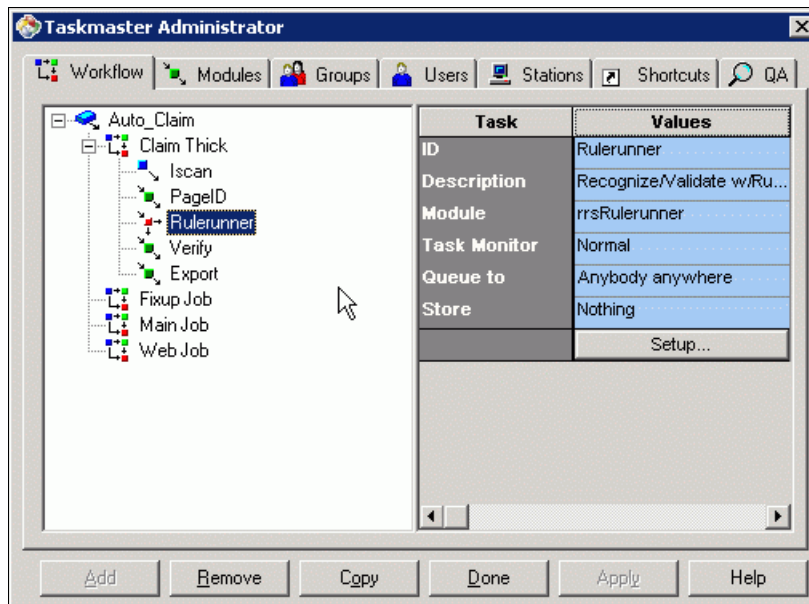


Figure 6-27 Setup of the Rulerunner job

For our use case, the Rulerunner task profile performs the following rule sets:

- ▶ Createdoc rule set
- ▶ Recognize rule set
- ▶ FindFingerprint rule set

The CreateDocs rule set

The CreateDocs rule set has two rules associated with it: Create Docs and Create Fields.

The Create Docs rule (Figure 6-28) contains one function. This function contains the CreateDocuments() action. This action creates the document structure. The action does not reorder the pages, but rather creates a document based on the page types.

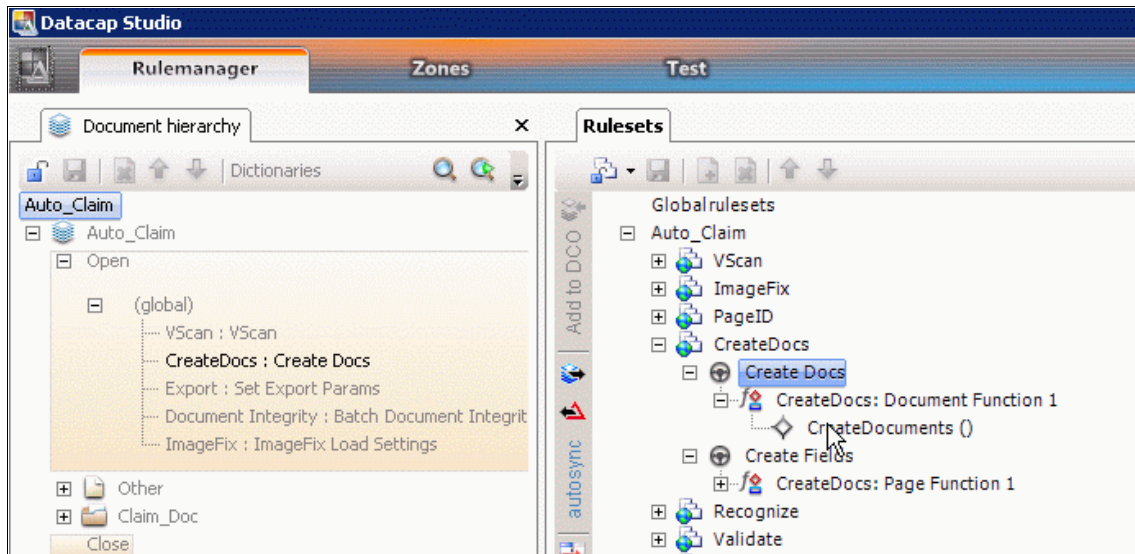


Figure 6-28 Create Docs rule of the CreateDocs rule set

The Create Fields rule contains one function. This function contains the CreateFields() action. This rule is attached to every page in the Document Hierarchy that has data fields attached to it. This action builds child fields on every page to which it is attached and that has fields defined on it. These fields all have a position value of "0,0,0,0" and contain no data.

Tip: This rule set normally does not require any additional work. You attach the Create Fields rule to each page that you expect to capture data field values.

The Recognize rule set

The Recognize rule set is used to perform a full page Optical Character Recognition (OCR). If no CCO is created yet, this rule set also creates the CCO file for the page. The CCO file is then used by other actions to locate data and find fingerprints. Because our use case has several pages, some of which we extract information from and fingerprint, we let this rule set create our CCO file.

To set up this rule set, make the following changes to the Recognize Page rule:

1. Click the **Recognize** rule set, and then click the **padlock** icon to lock it for editing (Figure 6-29). By locking the rule set, other users are prevented from changing it while you have it locked.

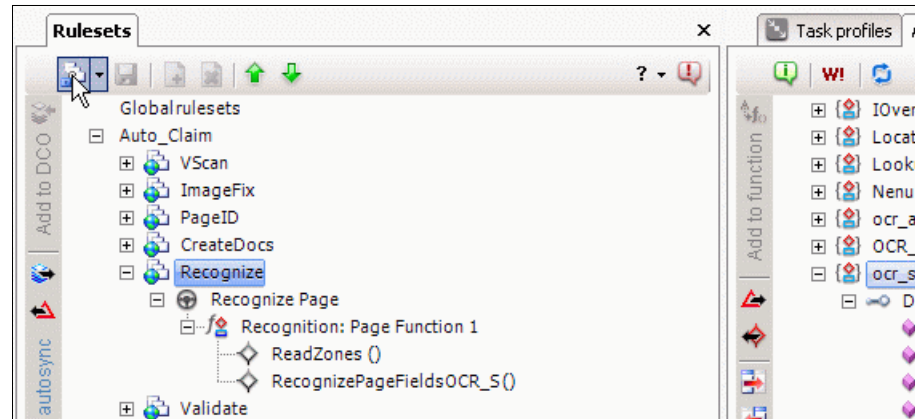


Figure 6-29 Locking a rule set and then adding a rule

2. Select a proper function:
 - a. Double-click the **Recognize Page** rule to open it.
 - b. Double-click the **Recognition: Page Function 1** function to open it.
3. Remove unwanted actions:
 - a. To remove the ReadZones() action, right-click it and then select **Remove**.
 - b. Do the same for the RecognizePageFieldsOCR_S() action.
4. Adding new actions.

To add the RecognizePageOCR_S() action to the function:

 - a. Click the function we want to add it to.
 - b. Click the action we want to add.
 - c. Click **Add to function**.
5. Click **Save** to save the rule set.

- Click the **down** arrow next to the **padlock** icon, and then select **Publish Ruleset** (Figure 6-30) to save the changes to the Recognize.rul file in the rules subdirectory of the DCO directory. The previous version gets added to the Recognize.rul.bak file.

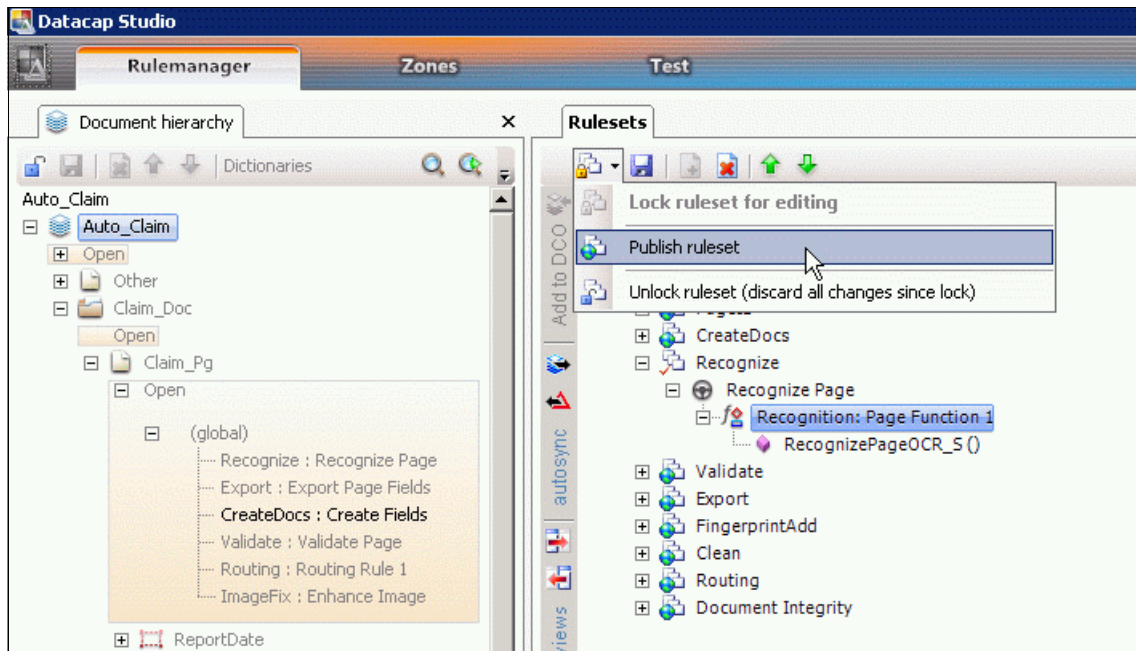


Figure 6-30 Saving and publishing the rule set

- Click the **down** arrow next to the **padlock** icon, and then select **Unlock ruleset**.

8. Add a rule to the pages that need a full page OCR (Figure 6-31).

We must attach the completed rule to the proper object in the Document Hierarchy:

- a. On the **Rulesets** tab, click the rule that you want to add.
- b. On the **Document hierarchy** tab, click the **padlock** icon to lock the Document Hierarchy.
- c. Expand the **Document Hierarchy** to select the object to which you want to add the rule. For our use case, we select **Claim_Pg**.
- d. Click **Add to DCO**.

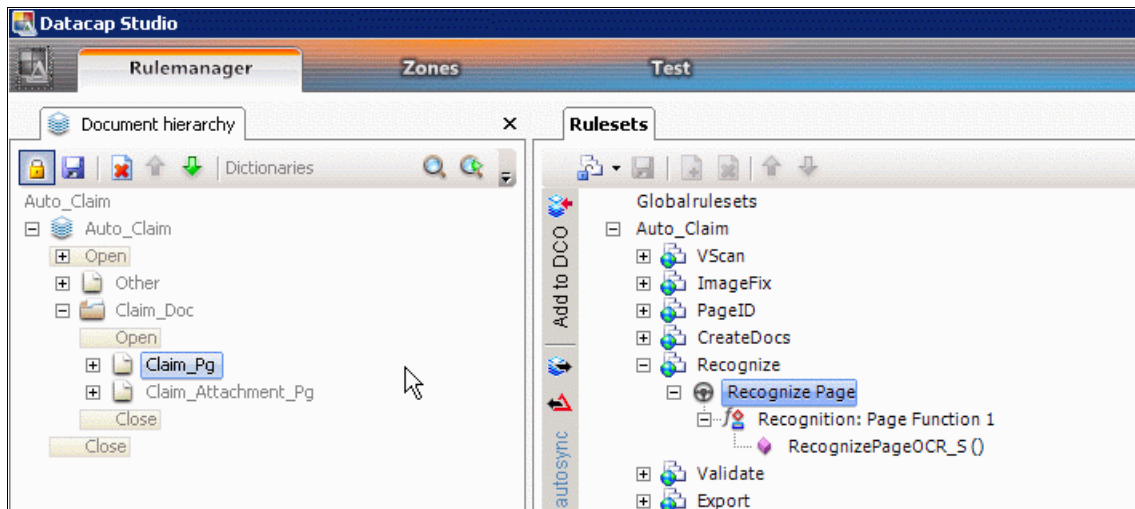


Figure 6-31 Applying a rule to DCO

If you receive the “Already exist” error message, expand the page object to ensure that the entry to the object is the “Ruleset: Rule” that you are trying to add. If there is a different pairing, look at combining the two rules.

For our use case, we initially received this error message because we were trying to add the “Ruleset: Rule” combination of “Recognize: Recognize Page,” and it already existed for that object.

Rule set entries per object: No more than one entry from a rule set can be attached to any given object.

The FindFingerprint rule set

With the Find Fingerprint rule set, we can compare the current runtime image with a list of known fingerprints. When it matches an existing fingerprint, it returns the value of the Template ID for the fingerprint to which it matches best. If it fails to match with any known fingerprint, depending on how the action is set up, it creates a fingerprint or does nothing.

The FindFingerprint rule set needs two rules to properly function, one rule with setup information and one rule to perform fingerprint matching. We use the FindFingerprint rule set to match the current runtime image with the ones we know of to find the best match.

Because this rule set does not exist, we must add a rule set first. Complete these steps:

1. Add a rule set:
 - a. In Datacap Studio, on the **Rulesets** tab, right-click the **Auto_Claim**, and select **Add New Ruleset**. You add a rule set to the bottom of the list. By default, it is called "Ruleset1." (If the name *Ruleset1* exists, the next incremental number is used.)
 - b. Rename the rule set to Find Fingerprint:
 - i. Click **Ruleset1** to select it. Pause and then click it again to invoke the rename option.
 - ii. Enter Find Fingerprint for the name.
 - iii. Click the **Rule1** rule, and rename it to Batch Level Fingerprint Settings.
2. Add the SetFingerprintDir() action to the "Batch Level Fingerprint Settings" rule:
 - a. Click **Function1**.
 - b. On the **Actions Library** tab, expand the **Autodoc** library.
 - c. Select the **SetFingerprintDir()** action.
 - d. Click the **Add to function** button to add this action to the function.
 - e. Enter "@APPPATH(fingerprint)" as the input parameter value for the action.

This parameter tells the system to look in the project folder at the project app file, and then find the entry for fingerprint and insert that value into this parameter. In our use case, we look in the \\morpheus\c\Datacap\Auto_Claim directory. In this directory, the system looks for the Auto_Claim.app file (the <project name>.app). This file includes an entry for the value for fingerprint, which is "fingerprint." This

action inserts the path variable “\\morpheus\c\Datacap\Auto_Claim\ fingerprint,” which indicates to the system that this directory is where the fingerprints are kept.

3. Add the SetProblemValue() action to the rule:
 - a. On the **Rulesets** tab, click the function level of the Batch Level Fingerprint Settings rule.
 - b. Under Actions Library, expand the **AutoDoc** library, and then click the **SetProblemValue()** action.
 - c. Click **Add to function**.
 - d. Set the input parameter to .75.

The SetProblemValue() action establishes the minimum acceptable value of a good match when comparing fingerprints. The higher you set this value, the more stringent the matching process is. For our use case, we set it to .75, which is a good starting point. Based on testing, you can adjust this number.

4. Add the SetFingerprintSearchArea() action to the rule, setting the input parameter to 0.5.

The fingerprints we match have CCO files that cover the entire page, and our runtime CCO covers the entire page. With the SetFingerprintSearchArea() action, you can focus on a particular portion of the image for the matching process.

For this use case scenario, we set this value to .5. Figure 6-32 shows the setup so far.

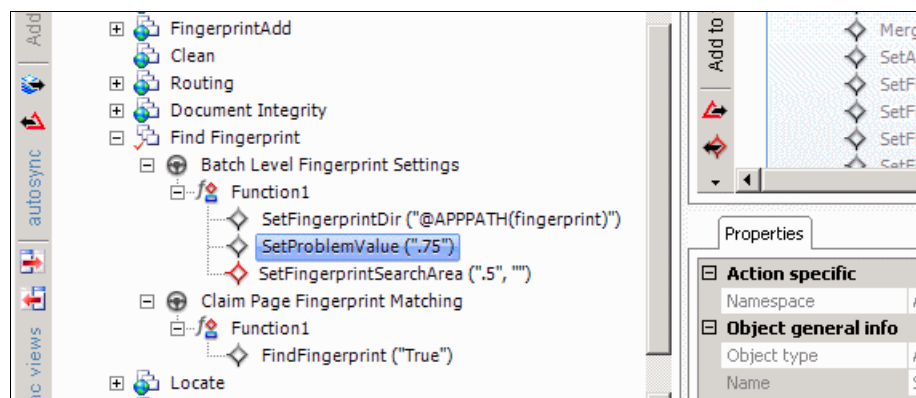


Figure 6-32 Fingerprint rule set

5. Add the Page Fingerprint Matching rule to the rule set:
 - a. Right-click the **Find Fingerprint** rule set, and select **Add Rule** to create a rule called Rule1.
 - b. Rename Rule1 to Claim Page Fingerprint Matching.
 - c. Under the rule, click **Function1**.
 - d. Under the Actions Library, expand **AutoDoc**, and then click **FindFingerprint**.
 - e. Click **Add to function**.

The FindFingerprint() action is now added to the function. The FindFingerprint() action uses the values that we set at the batch level and compares those values with the current runtime CCO. It looks at all of the CCO files and then selects the one with which it matches best.

If the action fails to match with any existing fingerprint, the FindFingerprint() action has one input parameter associated with it that tells the system what to do. The input parameter can be set to “True” or “False.” If the parameter is set to “True,” when the system fails to find a match, the system reports the closest match and uses the current CCO to create a fingerprint. The new template ID is assigned to the current page.

If the parameter is set to “False” and the system fails to find a match, the system reports the closest match but does not assign a template ID to the current page.

For our use case, we set the FindFingerprint(“True”) action for initial Fingerprint creation.

Tip: When initially developing a project, you can set the parameter for the FindFingerprint() action to “True” so that the system can learn and create new fingerprints. Later, when you are done with the development and you are now dealing with the structured form, you can set the parameter to “False” for the FindFingerprint() action.

Adding rules to the proper objects

Now that the rules are fully developed, associate the rules to the proper object in the Document Hierarchy. Although a rule can be associated to any object, we want to ensure that they are connected properly. We have two rules to attach, one to the batch level and one to the page level.

To add the rules to the proper objects, follow these steps:

1. Add the “Batch Level Fingerprint Settings” rule to the Auto_Claim batch object:
 - a. Click the **padlock** icon for the Document Hierarchy to lock it for editing.
 - b. Click the **Auto_Claim** batch object.
 - c. Click the **Batch Level Fingerprint Settings** rule.
 - d. Click **Add to DCO**.
 - e. Click the **Sync Views Left** button.

You now see an entry under the batch object called Find Fingerprint: Batch Level Fingerprint Settings as shown in Figure 6-33. This rule runs once for each batch, and it loads the batch-level fingerprint settings into memory.

We can load the fingerprint settings at any level down, including the page level. However, we must load the setting at the batch level to ensure that it is done only once for each batch and it reduces the loading time.

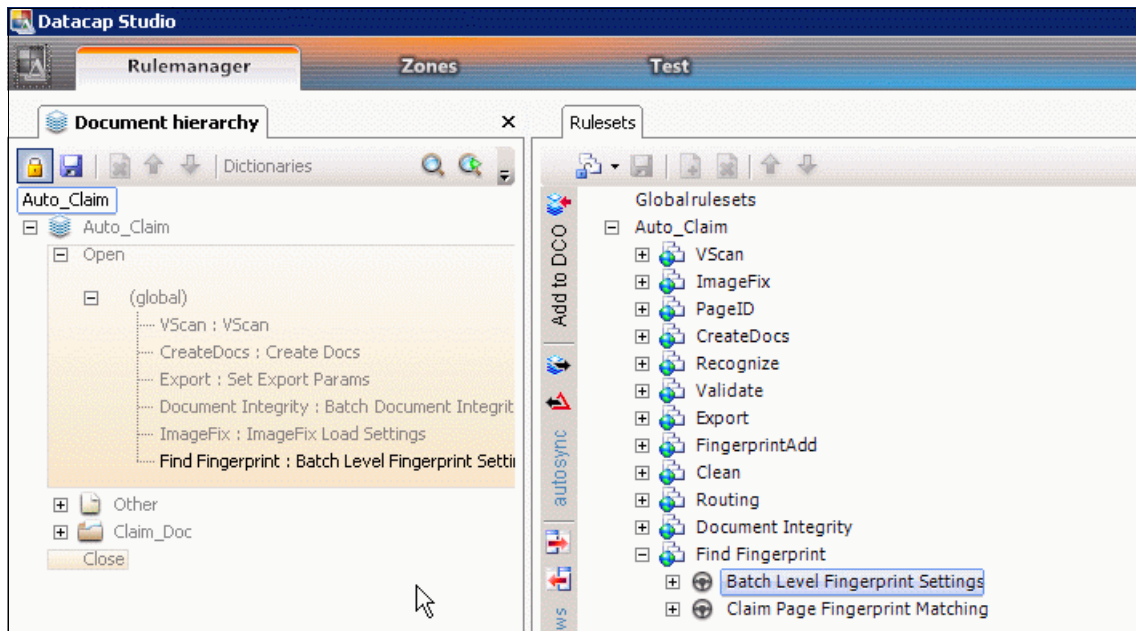


Figure 6-33 Applying rules to the DCO

2. Add the Claim Page Fingerprint Matching rule to the Claim_Pg object. Repeat the same step for Batch Level Fingerprint Settings rule, but add the rule under the Claim_Pg object this time.
3. On the **Document Hierarchy** page, click the **Save changes** icon to save the Document Hierarchy.

The saved changes are reflected in the <project>.xml file. In our use case, this file is the Auto_Claim.xml file, which is in the DCO directory.

4. Click the **Unlock DCO for Editing** icon to unlock the Document Hierarchy.

Adding a rule set to the Task Profile for Rulerunner

After creating the Find Fingerprint rule set and setting up the associated rules, add this new rule set to the Task Profile for Rulerunner:

1. Click the **Task Profile** tab.
2. Unlock the task profile.
3. Click **Rulerunner Task**.
4. On the **Rulesets** tab, click the **Find Fingerprint** rule set.
5. Click the **Add ruleset to profile** button, which adds the Find Fingerprint rule set to the bottom of the list for that task profile.
6. Click the green up and down arrows to place the rules in the order in which you want to run them. For our use case, we move the Find Fingerprint rule set immediately after the Recognize rule set.
7. Click the **Save** icon to save the task profile.

Removing unwanted rule sets

From this same location, we can also remove any unwanted rule sets. In our use case, we are not using the Document Integrity rule set. Therefore, we can remove it.

To remove a rule set, complete these steps:

1. On the **Task profile** tab, select the **Document Integrity** rule set to remove it.
2. Click the **Remove** icon.
3. Click the **Save** icon to save the task profile.
4. Click the **padlock** icon to unlock the task profile. This action saves the changes to the Administrator database and releases it for use.

The Document Integrity rule set is removed from the execution sequence. However, it is still listed under the rule set, and its rules are still connected to the object in the Document Hierarchy. Because the rule set is never called in the Task Profile, it never runs.

Figure 6-34 shows how the Task Profile looks now.

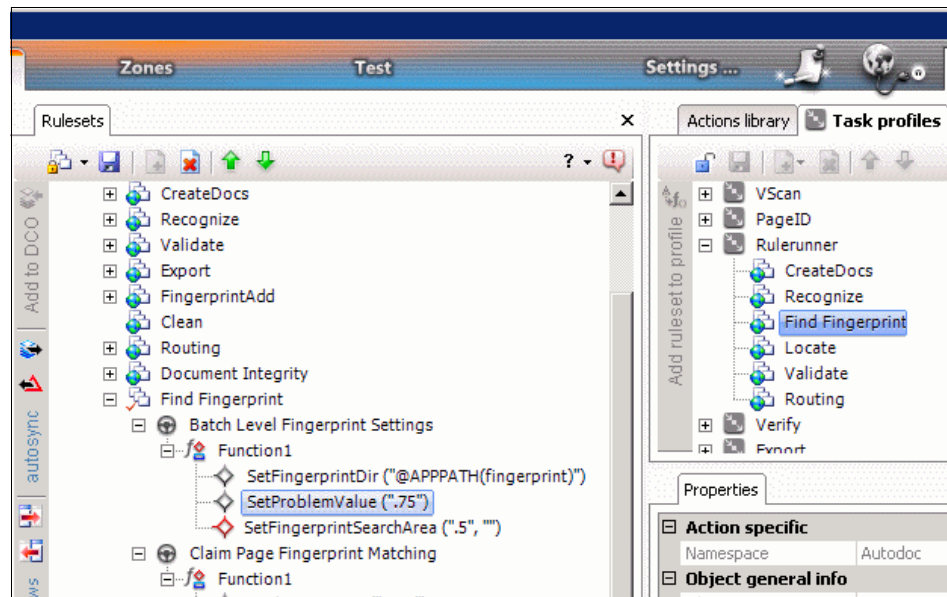


Figure 6-34 Task profile setup after adding the Find Fingerprint rule set

At this point, test the system. We have already scanned test batches to ensure the scanner works. If we move the scanned pages through the PageID and Rulerunner tasks, we must create a fingerprint in Datacap studio when we finish.

When adding new fingerprints, close Datacap Studio so that you can see the changes.

Important: Before closing Datacap Studio, verify that all rule sets have been saved and published, which is evident by the **blue globe** icon. Make sure that the Document Hierarchy and the Task Profiles are all unlocked.

6.1.9 Testing the progress

To test the application so far, complete these steps:

1. Scan in a new batch, and then let it run through both the Page ID and Rulerunner tasks.
2. Reopen Datacap Studio, and then select the **Auto_Claim** project.

3. In Datacap Studio, click the **Zones** tab. Click the **Fingerprints** tab, click **<New>** to expand it (Figure 6-35). You now see an entry with [Claim_Pg]. This entry has a sequential number to the left of it that is assigned to each new fingerprint that the system creates. In our use case, the number is 558.

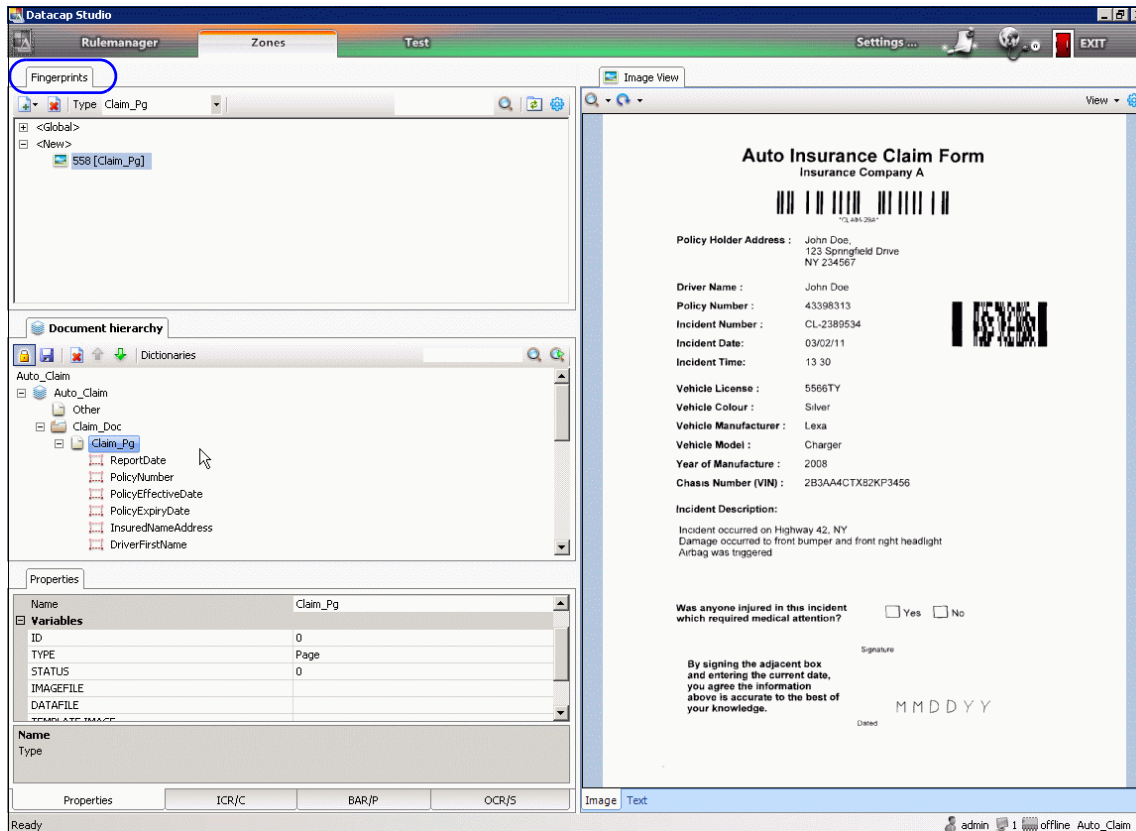


Figure 6-35 Adding a fingerprint

6.2 Zones and fingerprints

So far in our project creation, we set up rules to identify the pages and perform clean up on the images. We placed the pages into a document structure. We performed a full-page OCR. We ran the images through a fingerprint matching process and created a fingerprint.

Now we are ready to set the zones on the newly created fingerprint. Zones must be defined and located before validation on the content can be performed.

In Datacap Studio, on the **Zones** tab, click the newly created fingerprint, which is **558[Claim_Pg]** in this example (Figure 6-36).

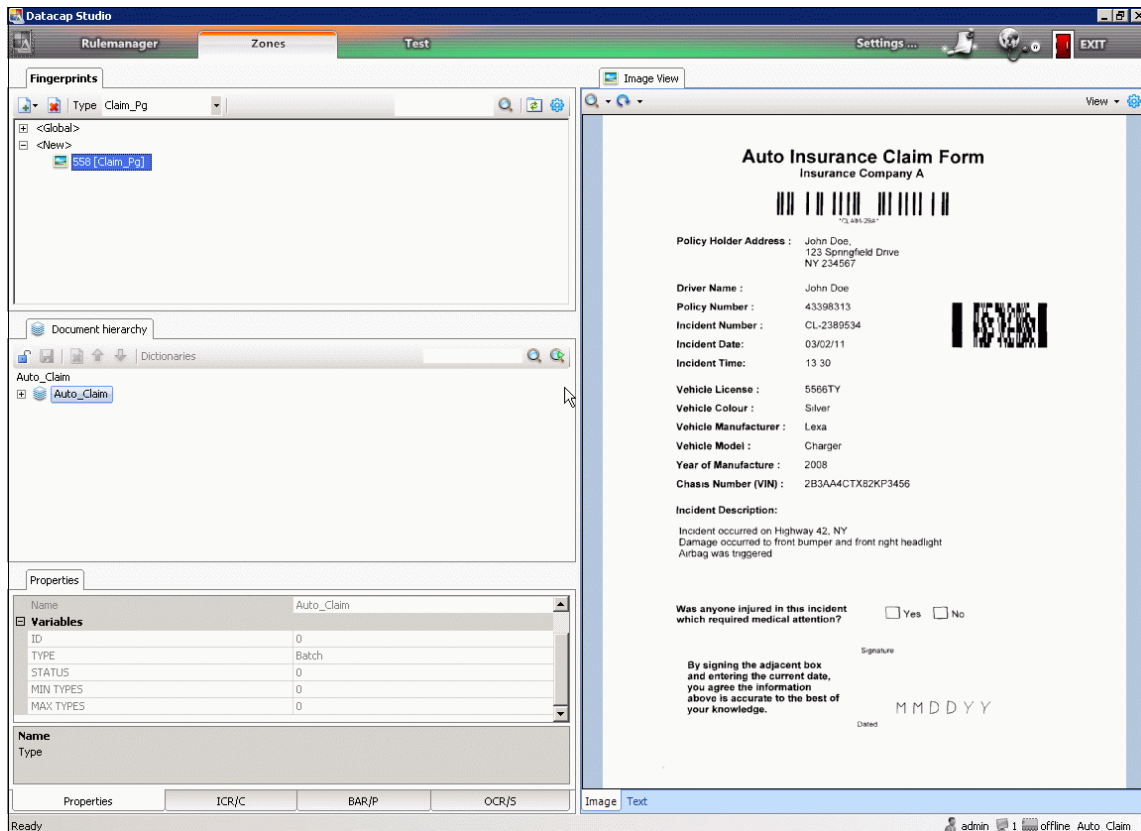


Figure 6-36 Newly created Claim_Pg fingerprint

In the right pane, you now see the image for the fingerprint that was just created. Looking at the image, you see that many of the vertical lines are gone. The reason why you no longer see these lines is because we arranged the PageID to come before the Image Enhancement rule set.

If you look in the batches directory, you see the TM000001.tif and TM000001.tio files. No .tio file exists for nonclaim sheet images because we only attached the Image Enhancement rule to the Claim_Pg page type. The .tio file is before the image enhancement version, complete with all lines and other image noise.

6.2.1 Setting up the zones

To set up zones, complete these steps:

1. On the **Fingerprints** tab, click the fingerprint for doing the setup.
2. On the **Document hierarchy** tab, lock the document hierarchy for editing.
3. Expand **Claim_Doc** → **Claim_Pg** (Figure 6-37).

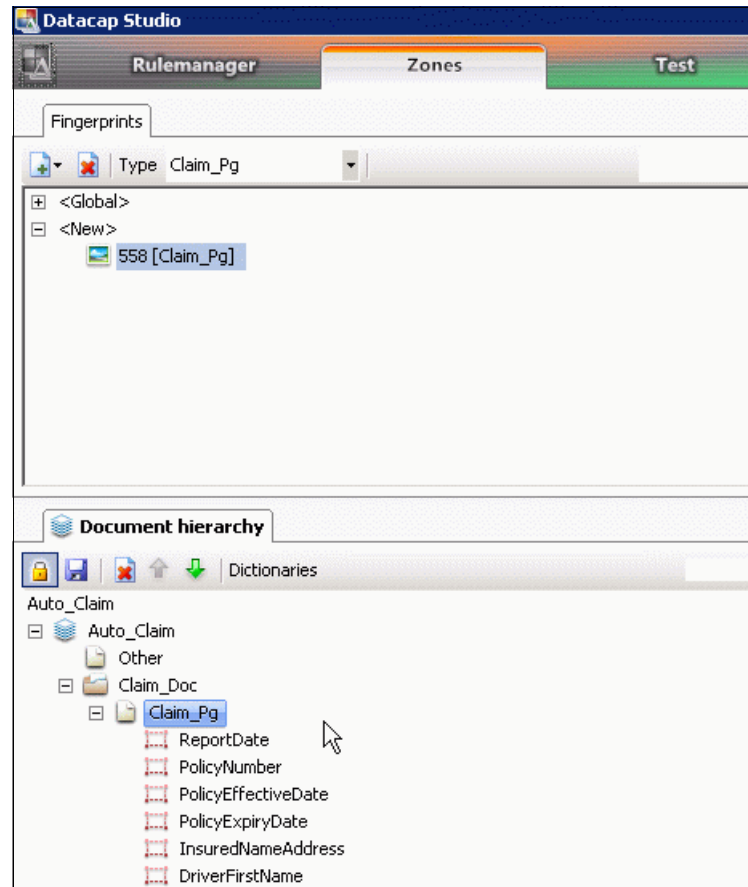


Figure 6-37 Claim_Pg zones setup window

6.2.2 Setting up optical mark fields

An optical mark (Optical Mark Recognition (OMR)) field has two levels. The first level is the coordinates for the entire field and is attached to the parent field. The second level is the coordinates at each box.

For our case study, in the Claim form, the claimant selects the Injury check box, indicating whether an injury was involved in the accident. Then the claimant signs the form. We want to set, as OMR fields, the Yes and No check boxes of the Injury field and the signature.

To set up the OMR fields for the fields, complete these steps:

1. Select the **Injury** field.
2. On the **Properties** tab, right-click the lower **Properties** tab. Right-click **Show tabs**, and then select the tabs to display or hide (for the tabs that you are not using; Figure 6-38).

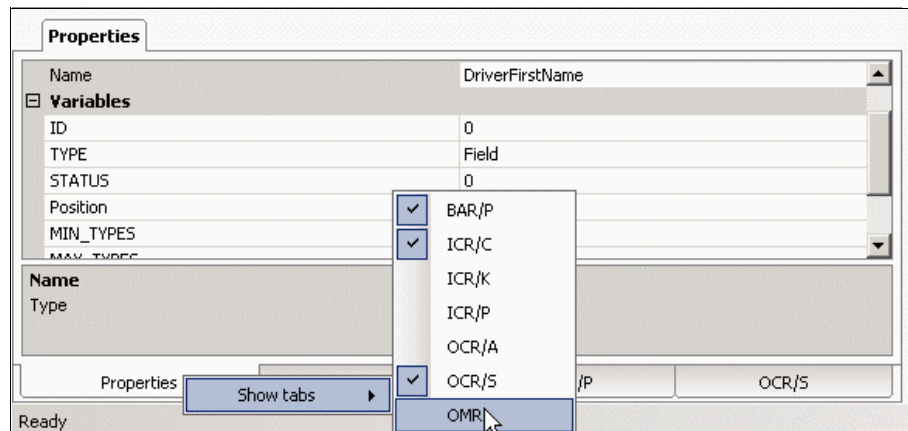


Figure 6-38 Selecting the OMR properties in Datacap Studio

Licensed options: Some choices on the **Properties** tab can be activated only through the purchase of additional licensing.

3. Set the OMR properties:
 - a. Click the **OMR** tab. The Properties view changes to reflect your choice.
 - b. Click the box next to the length, and then enter the number of check boxes that are available on the form. In the use case, because there are two boxes, we set the length to 2.

As soon as you set the length and tab out of the box, on the **Document hierarchy** tab for the Injury field, two new field boxes are displayed as the children fields to the field. These new fields are automatically named Injury_OMR1 and Injury_OMR2 (Figure 6-39). Do not rename these fields. The names are established by the system.

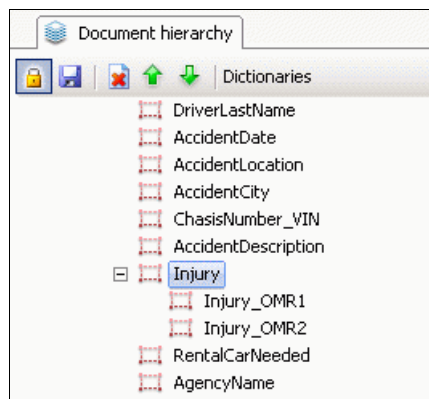


Figure 6-39 Assigning two OMR fields to the Injury field

4. Set the coordinates for the new OMR fields.

These coordinates are going to be associated with the view while in the Verify task. To set the coordinates, complete these steps:

- a. On the **Document hierarchy** tab, click the **Injury** field.
- b. Click the image on the right.
- c. In the upper left corner of what you want to show, while holding the left mouse button, drag down and to the right to set the lower right coordinate. When you finish, you see a dashed red box around the yes and no check boxes for the Injury question (Figure 6-40).

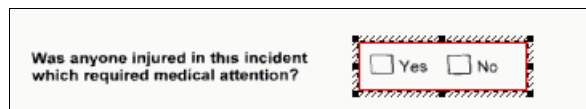


Figure 6-40 Setting up the zone around the Yes and No check boxes for the Injury field

- d. On the **Document hierarchy** tab, click the first of the child boxes, **Injury_OMR1**.

- e. On the image, draw another box that encapsulates the first check box value (Figure 6-41).



Figure 6-41 Setting up the zone around the Yes check box for the Injury_OMR1 field

- f. Repeat steps d and e for the Injury_OMR2 field, by drawing a box next to the No check box.
5. Repeat steps 1 on page 221–4 on page 222 for the InsuredSignature field. This time set the length to 1, because we are only looking to see if something is in the signature zone. Notice that, in this use case, in the first level box, we include the word *signature*, but in the second box, we exclude that word to avoid false hits. The word *signature* gives the verify operator a frame of reference for determining whether there is a signature there.

6.2.3 Setting up an ICR field

When you want Datacap to capture handwritten words or numbers, you set the fields as Intelligent Character Recognition (ICR) on the form. For the claim form, the claimant signs the form and writes the date when the claimant signs the form. The signature field is treated as OMR field. The date field, however, is processed as an ICR field.

To set up an ICR field for the SignedDate field, complete these steps:

1. On the **Document hierarchy** tab, click the **SignedDate** field.
2. On the image, draw a box from the upper left corner to the lower right corner, covering the entire area where the signed date will be written.
3. Click the **ICR/C** tab, and then set up the ICR/C Recognition Setup properties:
 - a. For the Recognition Type field, select **ICR Field**.
 - b. For the Remove Spaces field, select **Yes**.

Figure 6-42 shows the setup.

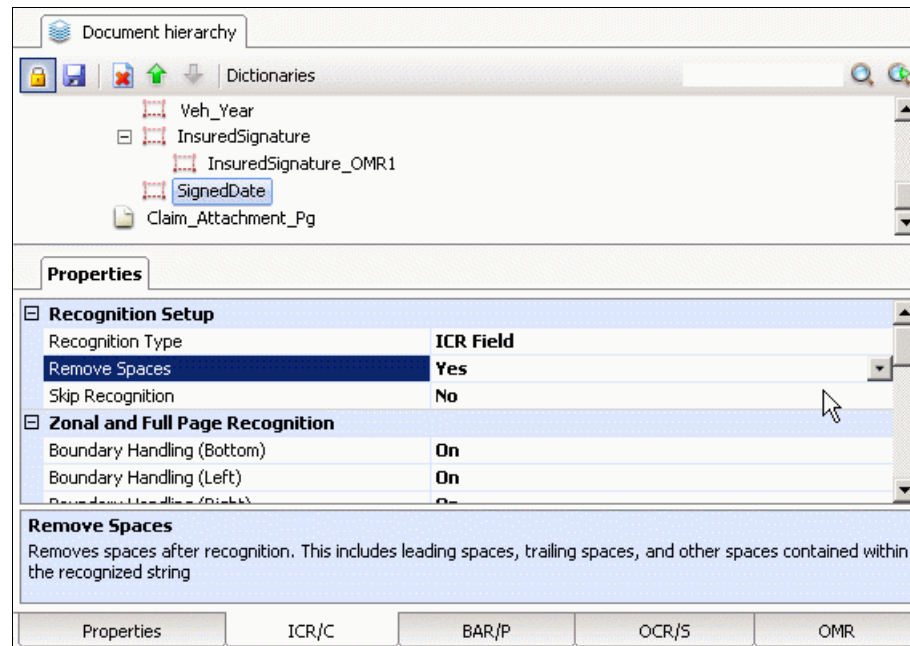


Figure 6-42 Setting up Recognition setup

4. On the **ICR/C** tab, set up the ICR/C Zonal and Full Page Recognition properties:
 - a. For the Character Set field, select **0-9**.
 - b. For the Classifiers field, leave it blank.
 - c. For the Country field, set to the specific country you want to use. For our case, we selected **USA**.
 - d. For the Length field, enter the number of characters you expect to find in that zone. For our use case, we set this field to 4.
 - e. For the Pattern field, enter a regular expression for the value you expect to find in the field. For our case study, we leave it blank.

Figure 6-43 shows the setup.

Properties	
Boundary Handling (Bottom)	On
Boundary Handling (Left)	On
Boundary Handling (Right)	On
Boundary Handling (Top)	On
Character Set	0-9
Classifiers	
Country	USA
Delete Intermediate project file	True
Dictionary	
Dictionary Mode	Incomplete
Font	Unknown
Length	4
Logical Context	Off
Number Of Lines	Unknown
Orientation	Normal
Pattern	
Pitch	Unknown

Pattern
The search result will match the specified RegEx-pattern.
Syntax

PropertiesICR/CBAR/POCR/SOMR

Figure 6-43 Setting up the Zonal and Full Page Recognition

5. Save the Document Hierarchy, and then unlock it.

6.2.4 Removing the Clean rule set

The Clean rule set is created by the system by default. Because we do not use it, we delete it from our project.

To remove the Clean rule set, complete these steps:

1. Go to the **Task Profiles** tab.
2. Lock the Task profiles for editing.
3. Expand the **Rulerunner** task, and then select the **Clean** rule set.

4. Click the **Remove** icon in the upper bar of this frame (Figure 6-44).

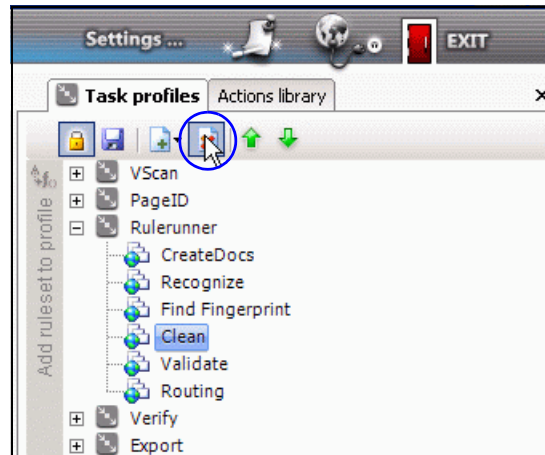


Figure 6-44 Removing the Claim rule set

5. Click the **Save** icon to save the changes.
6. Click the **padlock** icon to unlock the task profile.

6.2.5 Creating a rule set to read fields

We defined the zones on the page. Currently, they cannot be used because the system is not aware of them. Now, we must create a rule set that locates the previously defined zones and reads their contents from the captured form. We also need to assign the rule set to the DCO and Rulerunner task.

For our use case, in the claim form, we look for whether the document has a signature and date and whether the Yes or No checkbox was selected for Injury or if the field was left blank. As stated earlier, the Signature and the Injury fields are OMR fields. The date of the signature, the SignedDate field, is an ICR field. Therefore, in the new rule set, we must read both OMR fields and ICR field.

To create the rule set to read the claim form fields, complete these steps:

1. Go to the **Rulesets** tab.
2. Right-click the **Auto_Claim** rule set, and then select **Add ruleset**.
3. Rename the new Ruleset1 rule set to **Locate** and the Rule1 rule to **Claim_Pg Rule**. You can also rename the Function1 function. For our case study, we keep the same name for the function.

4. From the **Actions library** tab, add the **ReadZones()** action to the Function1 function. Figure 6-45 shows the current setup.

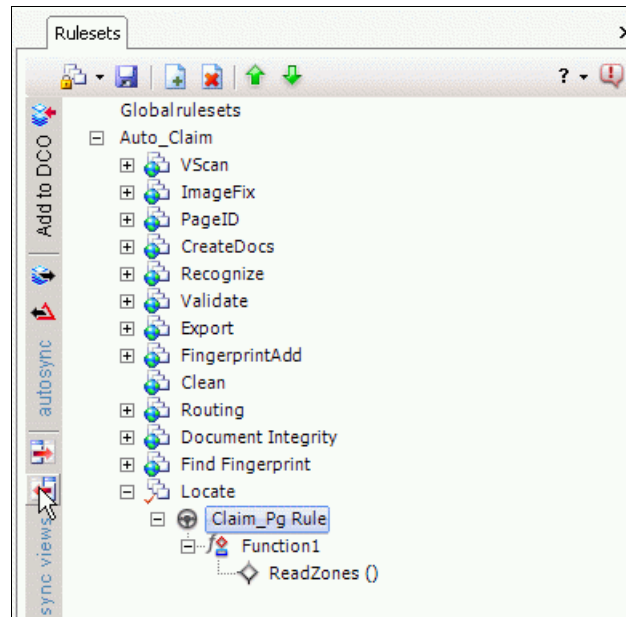


Figure 6-45 Adding the Locate rule set to the Auto_Claim rule set

5. Lock the Document Hierarchy for editing.
6. Select the **Claim_Pg** page, and then click the **Add to DCO** button.
7. Click the **left sync view** icon to verify that the correct rule has been applied to the right DCO.
8. Add a Read OMR Fields rule that reads OMR fields in the rule set (Figure 6-46 on page 228), and then apply the rule to the OMR fields in the DCO:
 - a. Select the **Locate** rule set, and then right-click **Add rule**.
 - b. Rename the new rule to ReadOMR Fields.
 - c. Select **Function1** under the ReadOMR Fields rule.
 - d. From the action library, add **RecogOMRThreshold()** to Function1.
 - e. Set the parameters for RecogOMRThreshold() to ("1", "0.5").
 - f. Apply the rule to the Injury and InsuredSignature fields in the DCO.

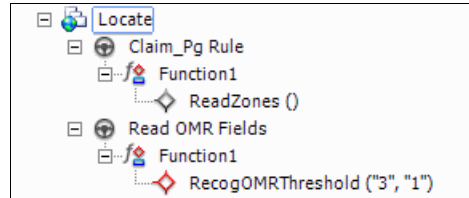


Figure 6-46 Read OMR Fields rule setup

To properly set the OMR value, see 9.2.7, “OMR field configuration” on page 298.

9. Add a Read ICR Fields rule that reads the ICR field in the rule set (Figure 6-47), and then apply the rule to the ICR field in the DCO:
 - a. For Rule name, enter Read ICR Fields.
 - b. For Function name, enter Function1.
 - c. For Action name, enter RecognizedFieldICR_C() (from the icr_c action library).
 - d. For Field applied to in DCO, enter SignedDate.

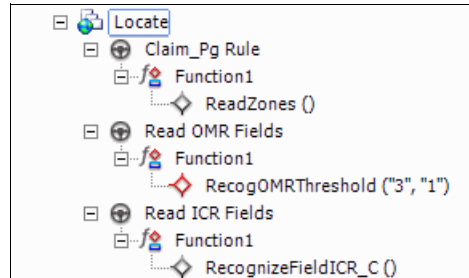


Figure 6-47 Read ICR Fields rule setup

10. Select the **Locate** rule set, and then save the changes.
11. Add the Locate rule set to the Rulerunner task profile. Move the rule set just after the Find Fingerprint rule set.
12. Save your changes.

Figure 6-48 shows the Locate rule set now.

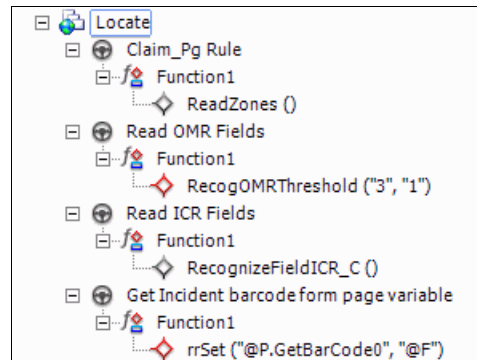


Figure 6-48 Locate rule set setup

6.2.6 Creating a rule set to validate fields

We bundle the validation of the content of the form in one rule set and name it “Validate.” This rule set connects to a DB2 database and performs a database lookup with the claim number (read from the barcode) as the key. The rule set then populates the fields on the claim page with the values from the database lookup. The additional content is then available on the form during the Verify step. At the end, the rule set checks whether the Injury, Signature, and SignedDate fields are completed and whether the SignedDate is in the correct format.

To create this rule set, complete these steps:

1. Select the **Validate** rule set, and then lock it for editing.
2. Verify that the application wizard added the Validate Page rule to the page object.
3. Add a validation rule for each field:
 - a. Right-click the **Validate** rule set, and select **Add Rule**.
 - b. Rename the new rule to Field populated from Database.
 - c. Add the `IsThisFieldFilled()` action to the function in this rule.
 - d. Add this rule to every field object. These fields are populated by the database lookup. These fields are different from the Injury, InsuredSignature, and SignedDate fields.
 - e. Add another rule to ValidatePage, and then name it OMR Field Marked.
 - f. Add the `ISMaxOMRChecked()` and `ISMinOMRChecked()` actions to the function.

- g. Apply this rule to the Injury and InsuredSignature fields.
- h. Add another rule to ValidatePage, and then name it SignedDate Field.
- i. Add the AllowOnlyChars() action to the rule with a parameter of 1234567890 to that rule.
- j. Add the IsFieldLengthMin() action with a parameter 6.
- k. Add the IsFieldLengthMax() action with the parameter 6.
- l. Apply this rule to the SignedDate field.

Figure 6-49 shows the final Validate rule set.

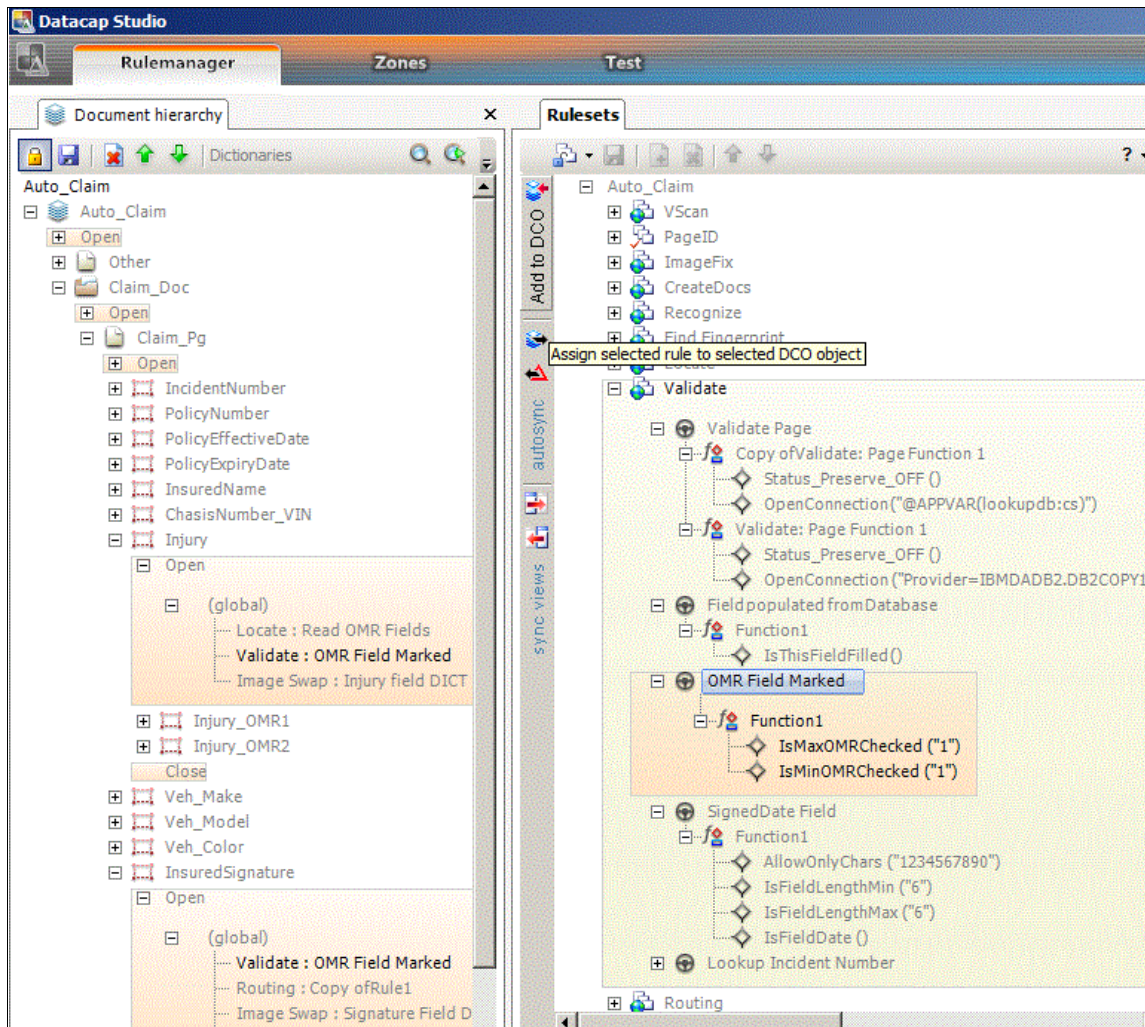


Figure 6-49 Validating the rule set

6.2.7 Validating captured field values against the database

The claim form has two fields that you can validate against the database, the form type field and the claim ID field. The form type field comes from the first barcode on the form. The claim ID field comes from the second barcode on the form.

The claim ID (the Incident_Number) is in the form of CL-#####, for example CL-2389534. In the database, the field value does not have the CL-. Therefore, to validate the claim ID against the database, we must strip away the characters and get the integer claim number, 2389534. Because the claim ID is created in the system when the accident is reported, there must always be a claim record matching with the claim ID.

With the claim record, we can get claim and policy information from the database, such as the policy expiration date. If the claim ID exists in the database, the validation goes through. We then populate fields from database to Datacap. If the claim ID does not exist in the database or is missing, we want to highlight the field in red color and mark it for operators to manually verify the field for the claim form.

Creating a validation rule for the Incident_Number

To create a validation rule for Incident_Number, complete these steps:

1. On the **Rulesets** tab, select **Auto_Validate**, and then click the **Lock for editing** icon.
2. Expand **Auto_Claim** → **Validate** → **Validate Page** → **Validate: Page Function 1**. Instead of a hard-coded user ID and password in `OpenConnection()`, enter `APPVAR(lookupdb:cs)` for the parameter string to look up the database connection information (Figure 6-50).

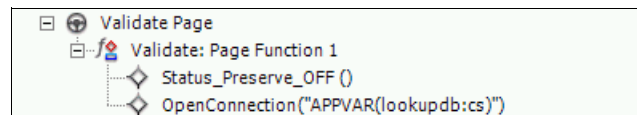


Figure 6-50 Validating the incident number: Validate Page Function 1

3. Add a rule for Lookup Incident Number as shown in Figure 6-51:

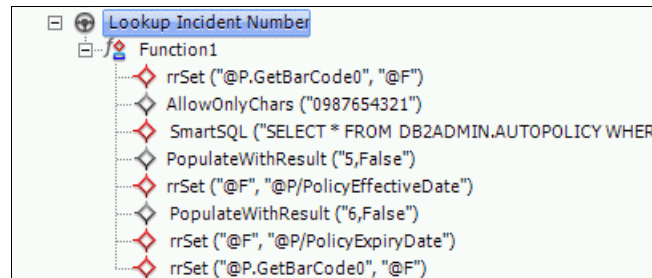


Figure 6-51 Adding a Lookup Incident Number rule

- Add the rule, and then rename it to Lookup Incident Number.
- Add an rrSet() action from the action library, and then set its properties as follows:
 - For varSource, enter @P.GetBarCode0.
 - For varTarget, enter @F.

This action reads what is scanned in the BarCode0 field, which is the claim number, and places the number in the current field, @F.

- Add the AllowOnlyChars() action from the action library and set its parameter as 0987654321.

This action reads the claim number from the current field @F, removing all characters, except characters 0–9. As mentioned earlier, in the claim form, the claim number has the format of CL-#####, for example CL-2389534. In the AUTOCLAIM database table, it is in digits, such as 2389534, without the “CL-” character.

- Add the SmartSQL() action from the action library, and then set its parameter as follows:
 - Set the first parameter:

```
Select * from DB2ADMIN.AUTOCLAIM C, DB2ADMIN.AUTOPOLICY P  
WHERE C.CLAIM_NUMBER=+@F+ AND C.POLICY_NUMBER=P.POLICY_NUMBER
```
 - Second parameter to YES.

This action runs the smart SQL as set in the first parameter. It passes in the CLAIM_NUMBER (stored in @F) from the previous action. Then it gets the claim record from the AUTOCLAIM table and the associated policy record from the AUTOPOLICY table.

The second parameter specifies whether you want the result set to return. In this case, we want to get the result set back for our case study to demonstrate that you can do that. In an actual production system, if you

need the information back, set it YES, but we advise that you return only the field that you need.

The result set returns rows as [...] [...] [...] The base index for the result set is from 1. To obtain the first field, you reference it as 1. To obtain the second field, you reference it as 2.

- e. Add actions to get fields from the result set, and then set them to the fields on the scanned page by using the following actions:

- `PopulateWithResult("#,<true/false>")`
- `rrSet("@F","@P/<field name>")`

The first action gets the *n*th field from the result set. If the field does not exist, exit the function. The second action sets the current @F field to the field of the page specified in *<field name>*.

For our example, we get and set the following fields:

- `PopulateWithResult("5,False")`
- `rrSet("@F","@P/PolicyEffectiveDate")`
- `PopulateWithResult("6,False")`
- `rrSet("@F","@P/PolicyExpiryDate")`
- `PopulateWithResult("1,False")`
- `rrSet("@F","@P/...")`

Getting the fields: You can tell what fields will return from the result set based on your database table schema. Alternatively, you can look at the log file to see what row set is returned from Datacap for testing and verification purposes. To see the log file, go to the machine that executed the batch scan. Go to the specific scanned batch directory, and then look for XML file under `yyyy#####.XML`, where ##### specifies the batch scan that was done.

- f. When all the fields are populated, add the `rrSet()` action to reset the current field back to the original field that was read by using `rrSet()`:

```
rrSet("@P.GetBarcode0","@F")
```

4. Attach the Lookup Incident Number rule to the field IncidentNumber in the scanned page:
 - a. From the **Document hierarchy** tab, expand **Auto_Claim** → **Auto_Claim** → **Claim_Doc** → **Claim_Pg** → **IncidentNumber** → **Open** → **(global)**.
 - b. Right-click **Validate (...was Field Populate ...)** (Figure 6-52), and then select **Delete** to delete the default rule set.

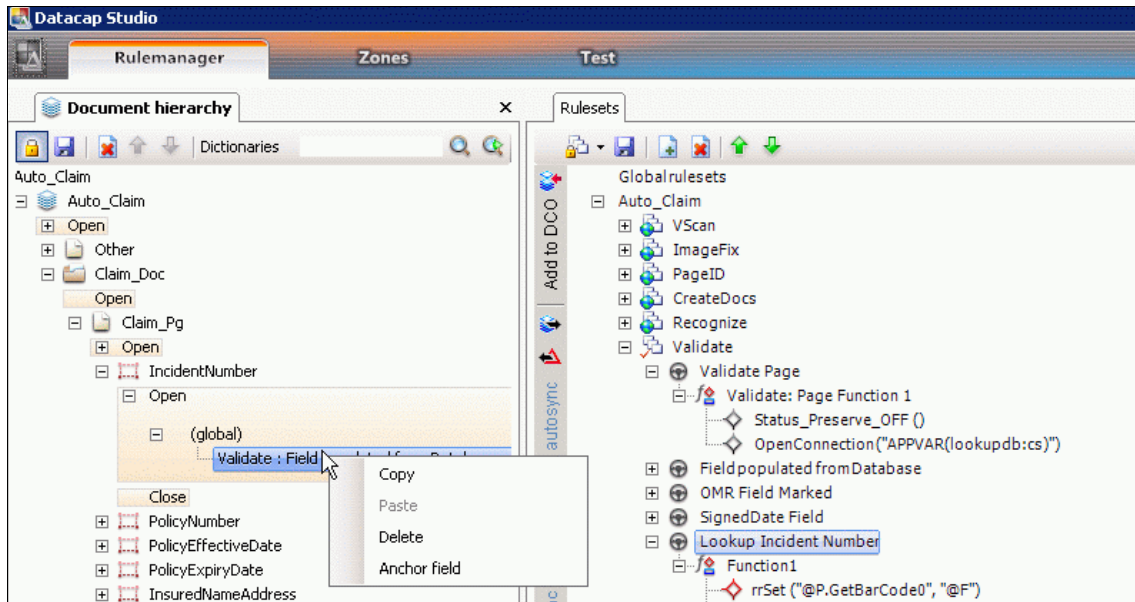


Figure 6-52 Attaching Lookup Incident Number to the page definition

You must delete the default validation rule set before you add the newly created rule set here. Otherwise, you receive an Already exists error message (Figure 6-53).

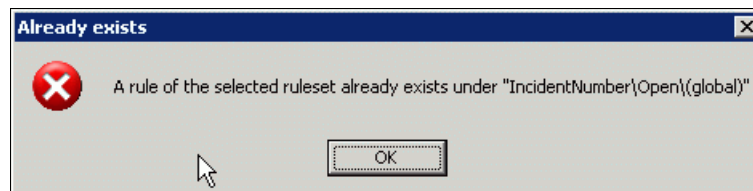


Figure 6-53 Error message when adding a second validation rule to the same DCO

- c. Apply the newly created Lookup Incident Number rule set to the DCO. From the **Rulesets** tab, click the vertical **Add to DCO** button on the left border of the Rulesets frame (Figure 6-54).

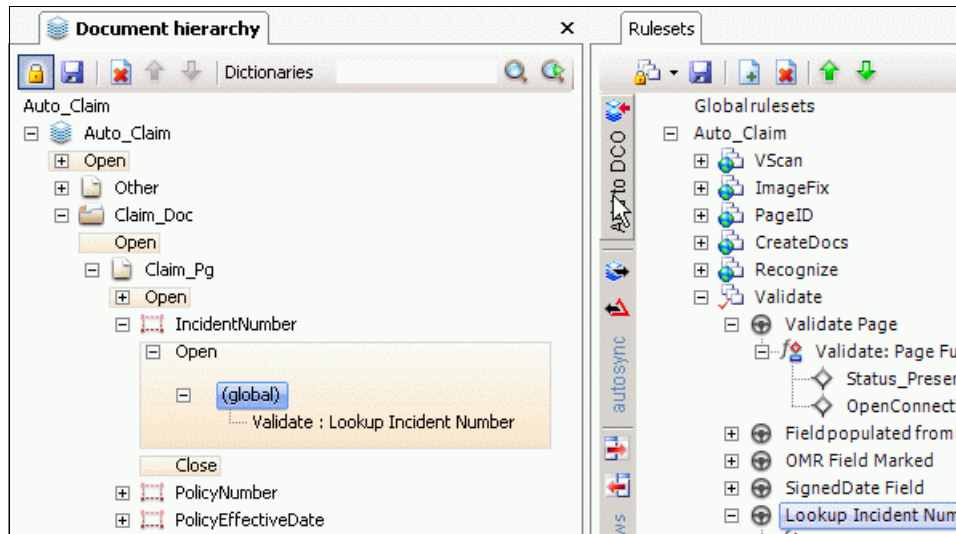


Figure 6-54 Adding the Lookup Incident Number rule set to the IncidentNumber field

5. Save all your changes:
 - a. On the **Rulesets** tab, click the **Save the change** icon to save the rule set that you created.
 - b. Click the **down arrow** icon, and then click **PublishRuleSet**.
 - c. On the **Document hierarchy** tab, click **Save Changes**.
 - d. Click the **Unlock DCO for Editing** icon.

6.2.8 Setting up routing

The Routing rule set decides the path of an image through the system. The rule set checks whether the confidence of the recognition is high enough to process the image automatically or whether an image must be routed for manual verification. In addition, it creates a separate image of the signature.

To set up the Routing rule set, complete these steps:

1. Go to the **Task profiles** tab, and then select **Rulerunner** → **Routing**.
2. On the **Rulesets** tab, expand **Auto_Claim** → **Routing** → **Routing:Rule1**.

3. Attach the routing rule to the claim page DCO in the Document Hierarchy frame with the following functions (Figure 6-55):
 - Function 1: ChkDCOStatus("1")
 - Function 2: ChkConfidence("8,1")

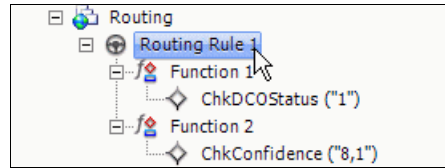


Figure 6-55 Check confidence for rule routing

The first action, ChkDCOStatus("1"), checks whether the DCO status is 1. We configure the system later, so that when the DCO status is 1, it means the rule set failed during the validation. When the validation fails, the image is not processed automatically.

The second function, ChkConfidence("8,1"), checks whether each character that has undergone OCR has a confidence weight of 8 or higher. For each character the scanner scans in, Datacap assigned a confidence number of 0–9 to it, with 9 being the highest. Therefore, 9 equates to 100%, and 8 means 90% confidence weight. If one of the characters that Datacap reads has a confidence weight lower than the specified number here, the system fails, and the document is not routed to other places. The operator must then do manual verification. The second parameter in the second function specifies whether to continue. For a value of 1, if any confidence weight of less than 8 for the character appears, stop the routing operation.

6.2.9 Testing scan validation and routing

Test the validation that is set up by scan in a document, and check the result by looking at the XML log file:

1. Launch the Taskmaster Client, and then log on to the Auto_Claim application, if you have not already done so.
2. Click the appropriate scan shortcut, such as the **iScan** icon to scan a new batch.

- Click the **Rulerunner** icon to run the batch through Rulerunner (Figure 6-56). The data file has been created.

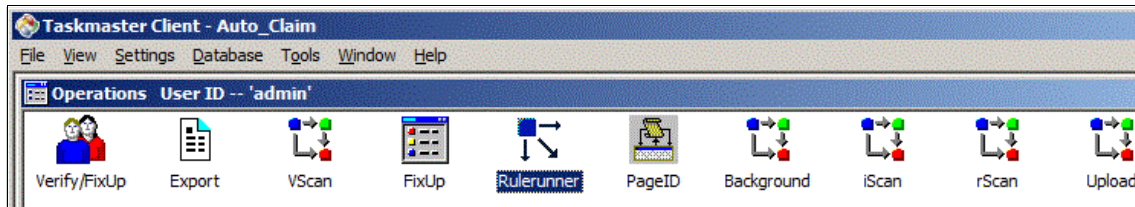


Figure 6-56 Running Rulerunner in the Taskmaster Client

- Go to the batch directory of your application on your Taskmaster Server. In this use case, we go to the `\\morpheus\c\datacap\auto_claim\batches\` directory.
- Open the current batch directory, for example `20110174.015\`, and then search for the `tm000001.xml` file (Figure 6-57)

F	InsuredSignature
	Text value : Y
	Char confi : 9
	TYPE : InsuredSignature
	Position : 811,1734,1563,1907
	STATUS : 0
	DensityString : 4
F	SignedDate
	Text value : 062511
	Char confi : 999999
	TYPE : SignedDate
	Position : 929,1984,1262,2048
	STATUS : 0
	IMAGEPATH :
	\\morpheus\c\datacap\auto_claim\batches\20110174.015
	\tm000001.tif
	RecogStatus : 0

Figure 6-57 Field data in the data file

An F in the first column indicates information about the field that follows it. The field name is given in the colored line. Information is displayed depending on the properties of the field. In this example (as shown in Figure 6-57), the SignedDate has a text value of 062511. Each character (in the SignedDate value) has a confidence of 9. This value means that the confidence of each character is 100%.

Depending on the type of the image you scanned in, you might see different results. Be sure to test an image with low confidence so that you see the routing stops.

6.2.10 Setting up the verification panel

You create the verification panel for the operator to work with. Batch Pilot creates a form and populates it with the fields in the page. With Batch Pilot, you can change the look and feel of the panel. For example, for the OMR fields, we want to display a meaningful word rather than a 0 or 1.

To set up the verification panel:

1. Select **Start** → **Programs** → **Datacap** → **Batch Pilot** → **Batch Pilot** to start Batch Pilot.
2. From the menu, select **Open Project**, and then navigate to the `dco_Auto_Claim` directory in your project directory to open the `rrs_verify.bpp` file.
3. In the lower frame, expand **Auto_Claim** → **Claim_Doc** → **Claim_Pg**.
4. In the column to the right, right-click **Claim_Pg** and select **Auto Form** (Figure 6-58 on page 239). This form reads the fields from this page and shows the snippets, which are a small extraction of the image, field label, and the data box. This form has just been created by Batch Pilot.

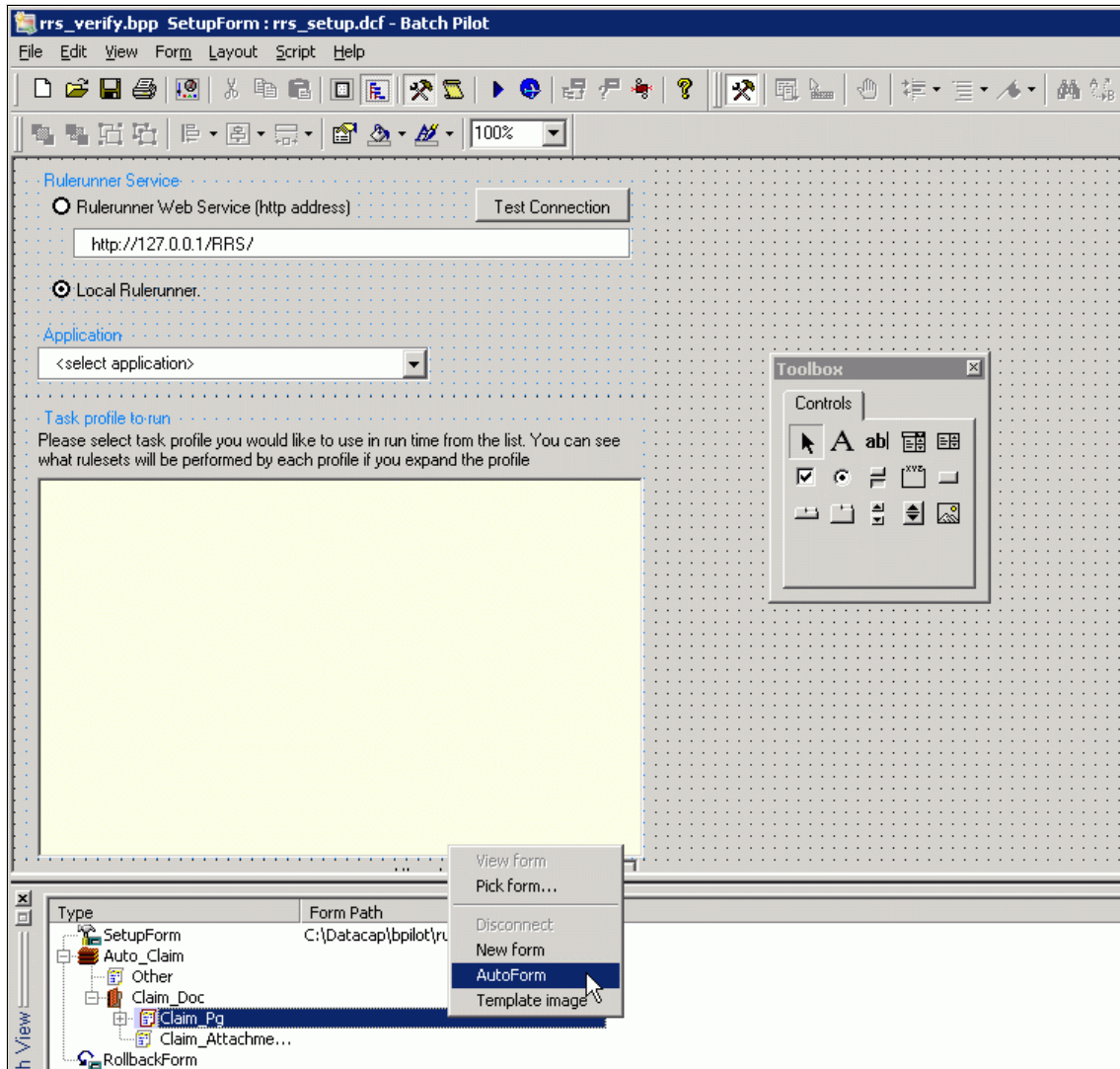


Figure 6-58 Creating an AutoForm for Claim_Pg

5. To begin configuring the form, select **File** → **Save Form** (Figure 6-59). Save this form in the dco_Auto_Claim\Verify directory of the project as Claim_Pg.dcf.

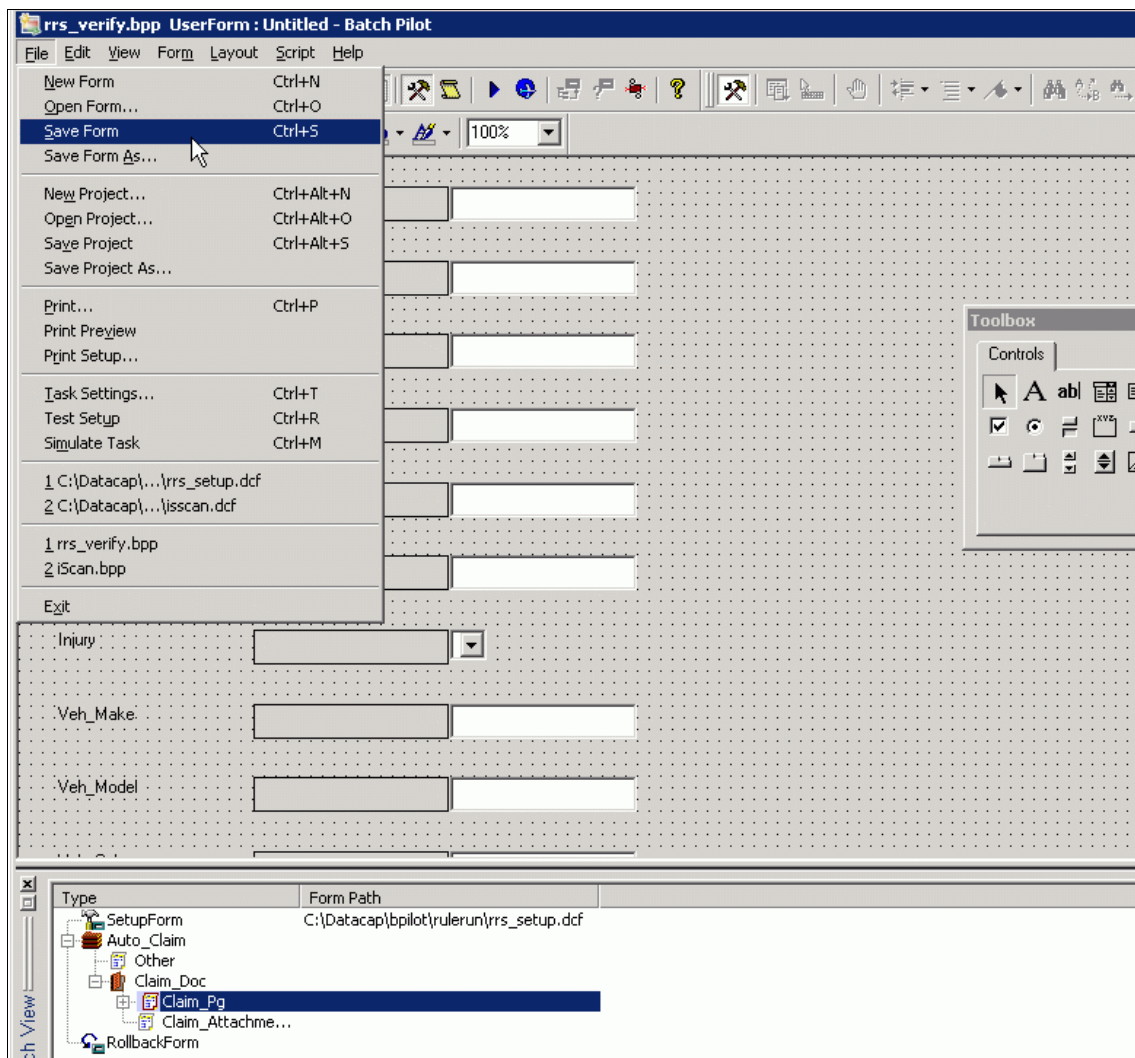


Figure 6-59 Saving the new form

- Go back to second column from the left, and go to the lower frame. Right-click **Claim_Pg**, and select **Pick form** (Figure 6-60). Navigate to the just saved form.

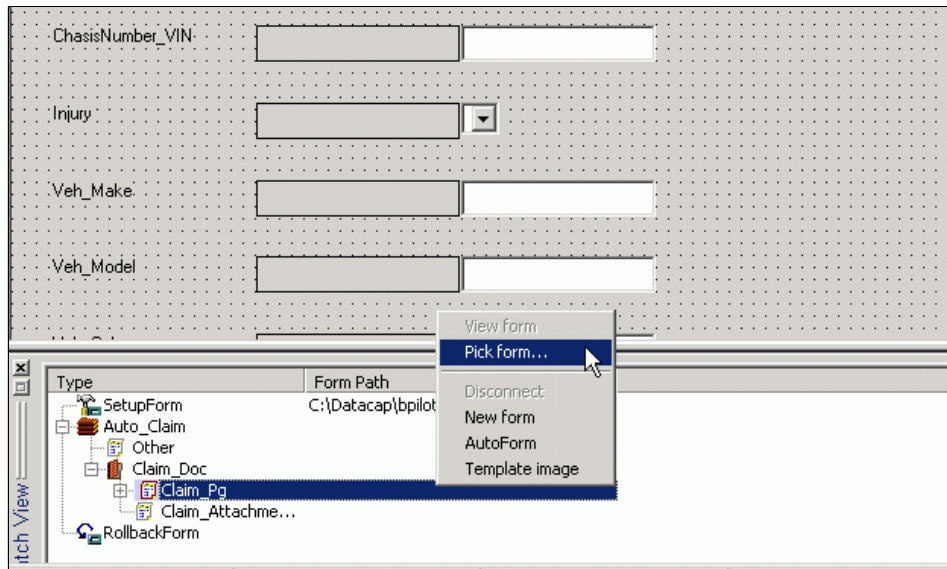


Figure 6-60 Picking the new form

- Open this **Claim_Pg.dcf** form.
- Save the project to make this form part of the project.

Now we must set up the Verify job in the Taskmaster Client.

6.2.11 Setting up the Verify job

To set up the Verify job, complete these steps:

- Launch the Taskmaster Client, and then click the **golden key** icon to open the Taskmaster Administrator window.
- In the Taskmaster Administrator window, click the **Workflow** tab.

3. Select the **Claim Thick** job, and select the **Verify** task. Then in the right frame, click the **Setup** button (Figure 6-61). Batch Pilot opens.

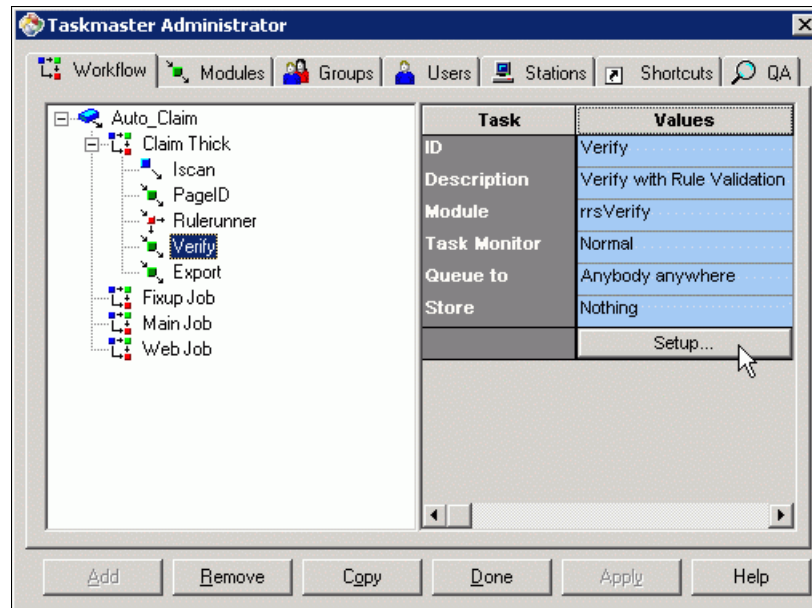


Figure 6-61 Opening the Verify task

4. Select **File** → **Task Settings** (Figure 6-62).

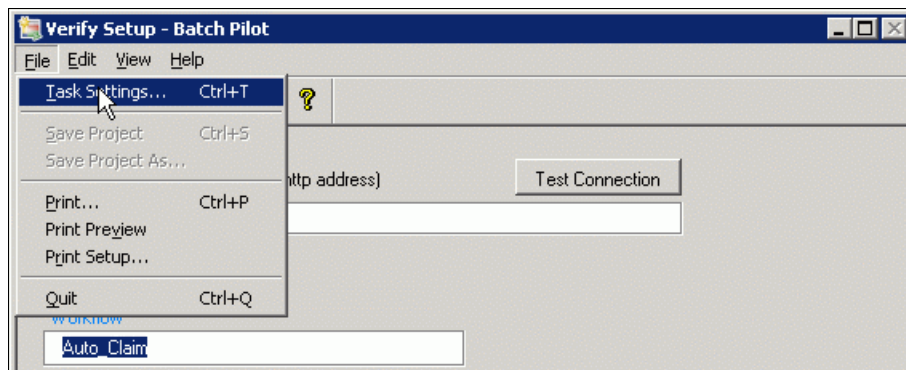


Figure 6-62 Selecting Task Settings

5. In the Settings for Verify task window (Figure 6-63), complete these steps:
 - a. Click the **Filters** tab.
 - b. For the Level field, select **PAGE**.
 - c. For the Property field, select **STATUS**.
 - d. For the Type field, select **Claim_Pg**.
 - e. For the Problem value, enter 1.

Confidence level for page type: We want to present a page of type Claim_Pg to the operator, only if the confidence is below a defined level. In this case, the Status was set to 1 by the validation rule.

- f. Click **Add**.
- g. Click **OK** to save the changes.

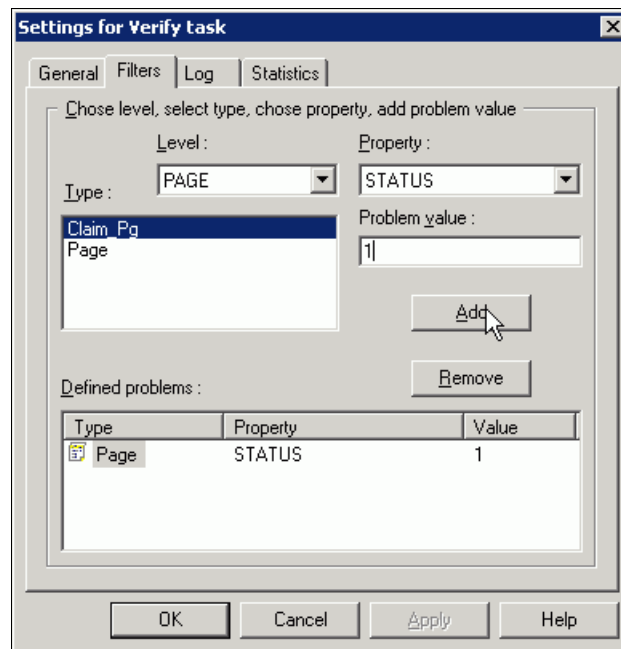


Figure 6-63 Task settings, add problem value

6. In the Taskmaster Administrator window, click **Done** to publish the changes.

We can test the verification panel in the Taskmaster Client now. However, first, we want to translate the boolean values of the OMR fields to more values that are easier to use. To achieve this goal, we create a dictionary within Datacap Studio.

Creating a dictionary for value translation

To create the dictionary for value translation, complete these steps:

1. Go back to Datacap Studio.
2. Navigate to **Rulemanager** → **Document hierarchy**.
3. Lock the **DCO** for editing, which changes the **Dictionaries** button in the icon row of the **Document hierarchy** tab from gray to black (Figure 6-64).

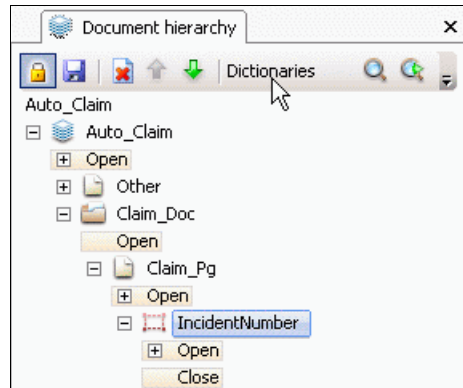


Figure 6-64 DCO: Manage dictionaries

4. Click **Dictionaries**.
5. In the window that opens and shows the available dictionaries, click the **Edit Dictionary** icon on the far left side. Hover over this icon, and then select **Add dictionary** (Figure 6-65).

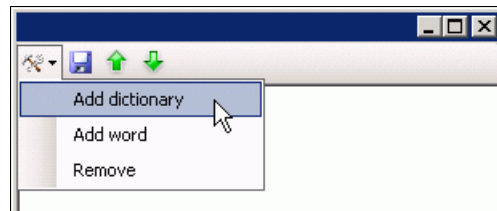


Figure 6-65 Adding a dictionary

6. Give the new dictionary the name YesNo, which is done the same way as you change the name of a function.
7. Right-click the new dictionary, and then select **Add word**.
8. Rename <new word> to Yes and “value” to Y.

9. Add another word and value pair, where you rename <new word> to No and “value” to N (Figure 6-66).

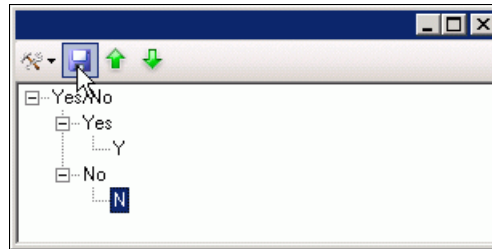


Figure 6-66 Adding the word and value pairs

10. Save the DCO. Do not unlock it yet.
11. With the DCO still locked, right-click the **Injury** field, and select **Manage variables** (Figure 6-67).

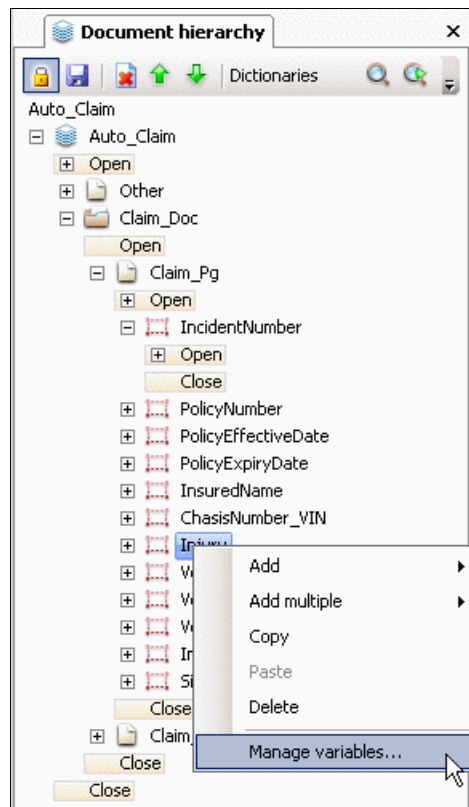


Figure 6-67 Managing field-level variables

12. In the Injury pane (Figure 6-68), in the upper left corner, select **New**. Then enter the var field level variable name of DICT. Keep in mind that this name is case sensitive.

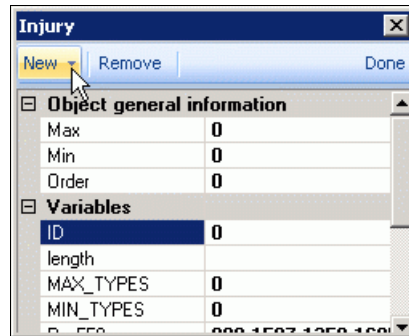


Figure 6-68 Attaching a dictionary to a field

13. Name this variable Yes/No. Then click **Done**.
14. Save your changes, and then unlock the DCO.
15. Go back to the verify window.

Now, a dictionary has been created and attached to the Injury field. From now on, a field of value 0 is translated to No, and a value of 1 is translated to Yes. Users on the verification panel no longer see the technical return value of the OMR function. They see a returned value that is closer to what is displayed on the image.

6.2.12 Setting up the export to the repository

We decide to export the multipage TIF files into the repository. This way, an image closest to the original is stored in the repository for further processing and archiving.

During the process, the original image, which was created by the scanner, has been renamed to <filename>.tio. The processed images are split into single paged images and named <filename>.tif. We must prepare the files for export to the FileNet repository. We want to export one file for each document to the repository. The file must have the .tif extension for it to display in the client of the repository.

To create the Image Swap rule set, complete these steps:

1. Add the `RenameFile()` action to the function `Function1` in the rule.
2. Rename the `Claim_Pg.tif` file in the Image Swap rule set twice. The first call changes the extension from a `.tif` file to a `.tip` file. The second call changes the extensions from a `.tio` file to a `.tif` file.
3. Attach this rule to the `Claim_Pg` DCO.
4. Create the Make Multipage tif rule.
5. Add the `TifMerge_SetFileName()` action to the function `Function1` in that rule, which sets the filename to `<DocID>.tif`.
6. Add the `TifMerge_MergelImages()` action to the function `Function1` in that rule. With the `All` parameter, this action merges all images into the multipage image. In our use case, it adds the snippet of the signature and it has been saved earlier.
7. Attach this rule to the Close event of the `Claim_Doc` DCO.

Overwriting the OMR field values with the values in the dictionary

Before exporting to the FileNet P8 system, we want the OMR field values from the dictionary rather than a 0 or 1. To overwrite the field values with the values in the dictionary, complete these steps:

1. Create the Signature Field DICT rule. The default `Function1` function is created. Add the `rrSet()` action to set the `InsuredSignature` field.
2. Create the Injury Field DICT rule. The default `Function1` function is created. Add the `rrSet()` action to set the `Injury` field.

Figure 6-69 shows the current setup.

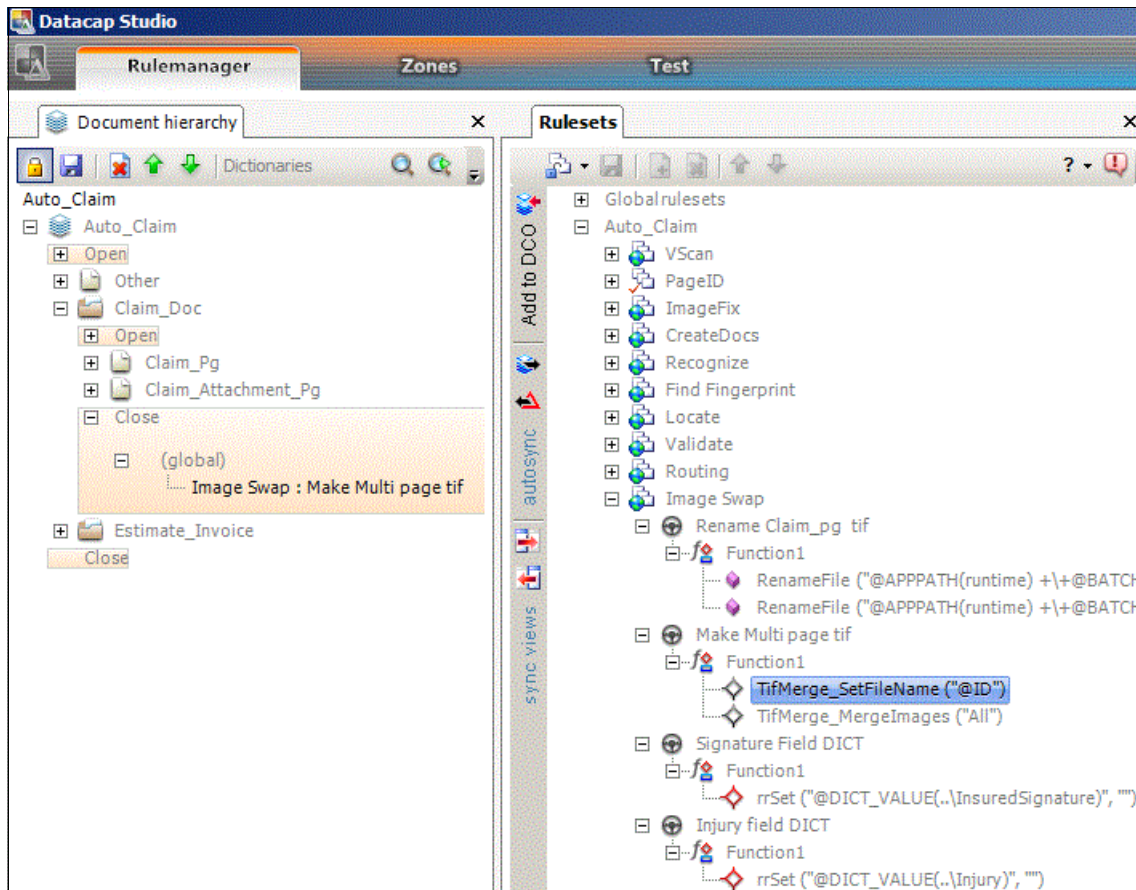


Figure 6-69 Overwriting the field values with the values in the dictionary

Now we can create the rule set to export the document to FileNet.

Exporting the document to the FileNet P8 system

To create the Export out to FN rule set, complete these steps:

1. Add the FN Setup values rule to that rule set.

In the FN Setup values rule, the following actions are located to connect and log in to the FileNet repository. There is no data transfer yet.

- FNP8_SetURL() to set the URL to the repository
- FNP8_SetTargetObjectID() to set the object ID in the FileNet repository
- FNP8_SetTargetClassID() to set the class ID in the FileNet repository
- FNP8_Login() to log on to FileNet (This action logs on to FileNet.)

2. Assign the rules to the open event of the batch.

In our case, these actions are general actions for the whole batch. Therefore, the FN Setup values rule set is assigned to the batch (Figure 6-70).

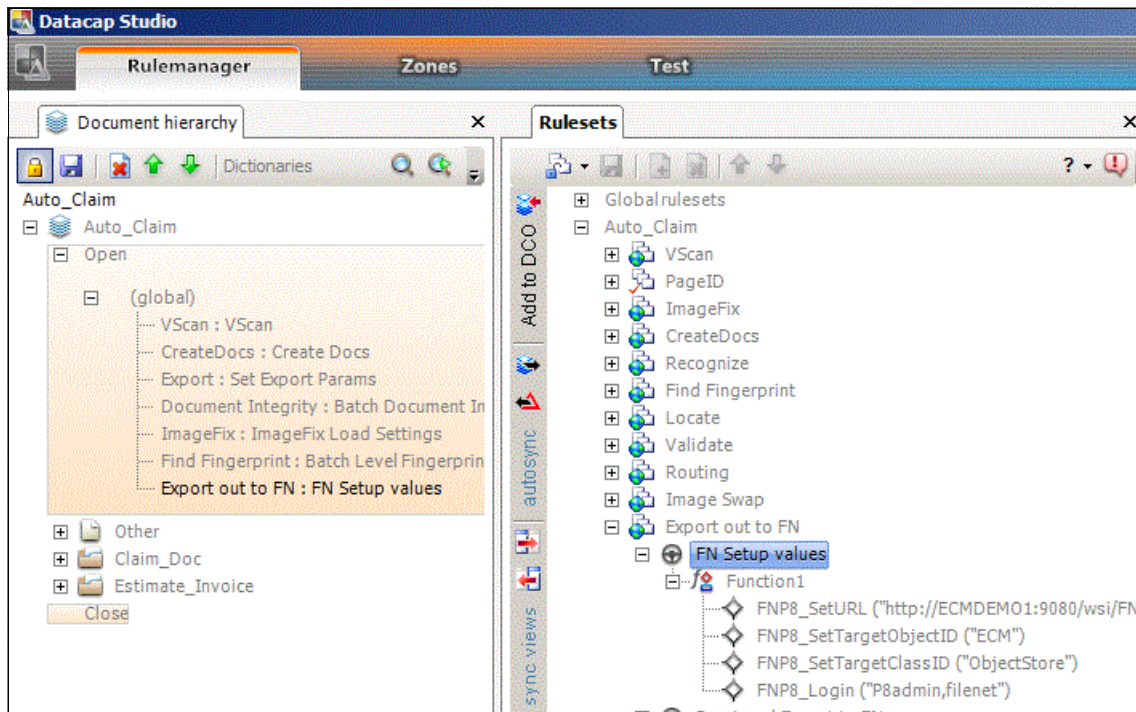


Figure 6-70 FN Setup values rule set assigned to DCO

3. Create the Doc Level Export to FN rule for the Export out to FN rule set. It contains the following actions:
- FNP8_SetDocClassId() to set the document class ID in the FileNet repository
 - FNP8_CreateFolder() to create a folder within the FileNet repository
 - FNP8_SetProperty() to set a property in the FileNet repository (called once for each property to be set)
 - FNP8_SetDocTitle() to set the document title in the FileNet repository
 - FNP8_Upload() to upload the image to the FileNet repository

4. Assign the rule to the open event of the document (Figure 6-71).

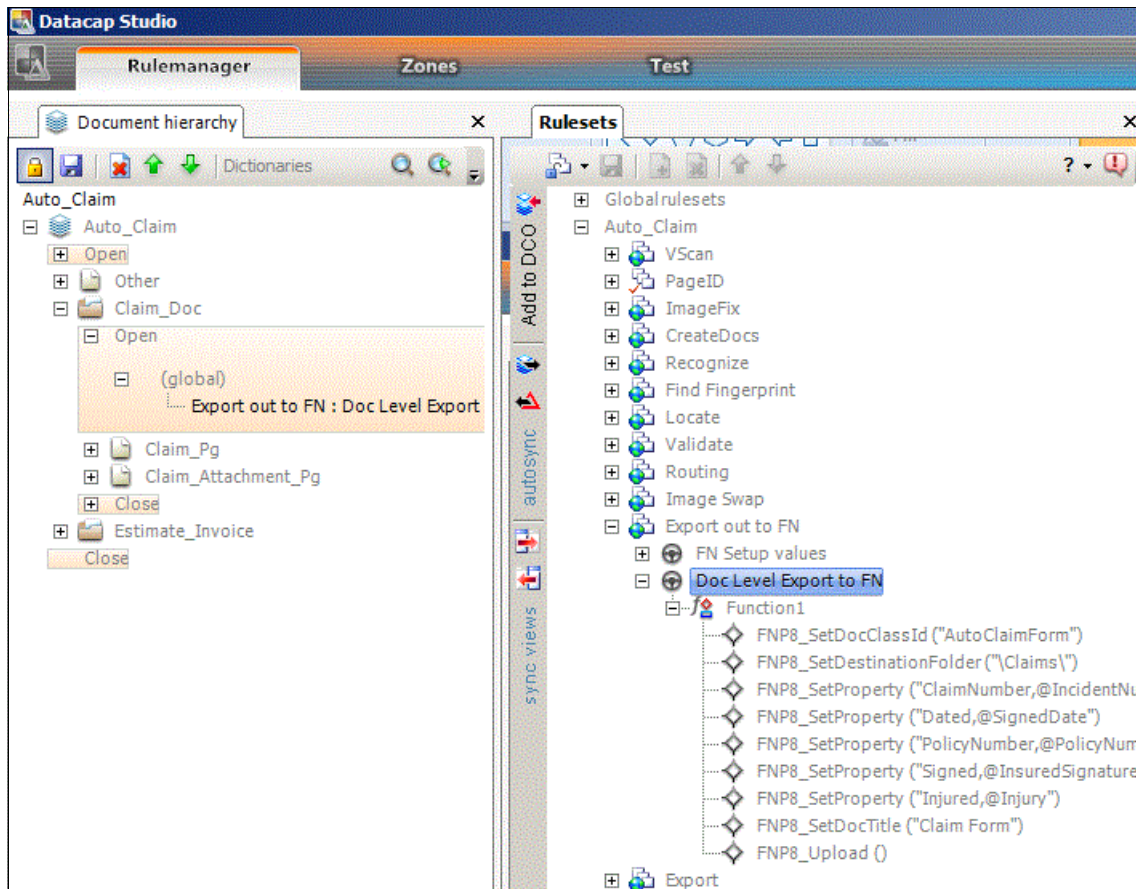


Figure 6-71 Export out to FN rule assigned to DCO

After a successful upload, the P8_Uploaded.xml file is created in the batch directory.

P8_Uploaded.xml file: If you debug this functionality, be sure to delete the P8_Uploaded.xml file before each run. The existence of this file prohibits another successful export.

6.3 Task profiles overview

We have created an application that serves our use case. Figure 6-72 provides an overview of the Taskmaster task profiles.

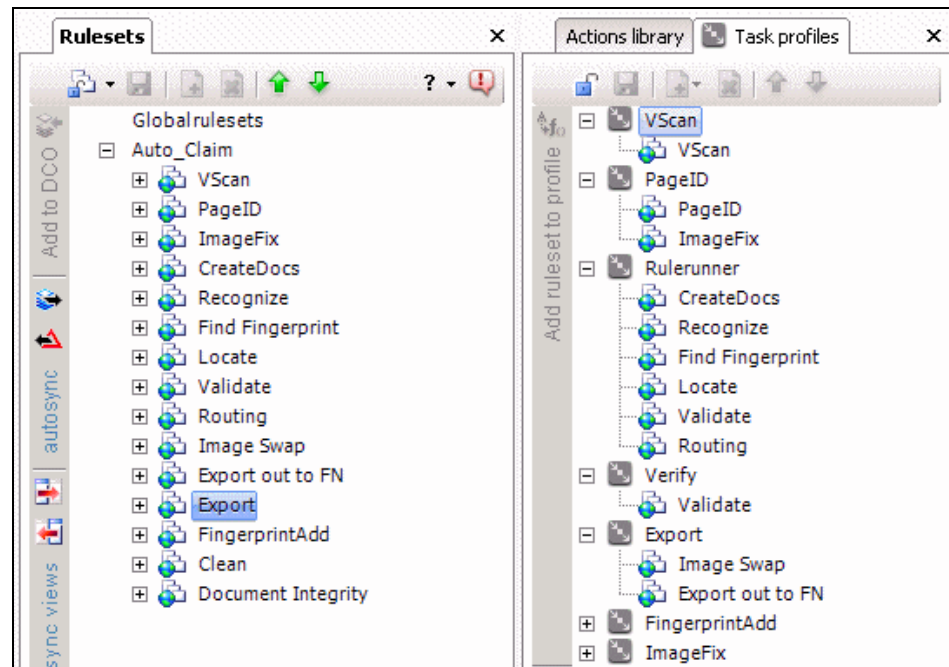


Figure 6-72 Task profiles overview

Task profiles sequence: The sequence of the tasks on the **Task profiles** tab is important. The tasks are run from top to bottom and are run on each DCO to which they are assigned.

The order of the rule sets in the **Rulesets** tab does not affect the execution order. However, as a good practice, sort them in a meaningful order.



Adding a document type to an existing application

This chapter explains how to add a document to an existing Datacap application. It also explains how to validate the fields within the document, routing, creating verify panel, and exporting.

This chapter includes the following sections, which outline the process for adding documents, specifically the Estimate and Invoice documents, to an existing application:

- ▶ Adding VScan to a rule set
- ▶ Adding a document with pages and fields
- ▶ Setting the PageID logic
- ▶ Adding the Image Enhance feature to the pages
- ▶ Configuring the CreateDocs rule set for the pages
- ▶ Adding a full page OCR rule set to the pages
- ▶ Setting the fingerprint for the Estimate_Invoice document
- ▶ Obtaining field values in the Claim_Pg document by using the Locate() rule set
- ▶ Looking up vendor information from a database by using the Lookup() rule set
- ▶ Validating data
- ▶ Routing scanned pages
- ▶ Verify task with Batch Pilot
- ▶ Export task

7.1 Adding VScan to a rule set

To set up the VScan task, complete these steps:

1. In **Document hierarchy** tab, expand the application. In this example, we expand **PIE_RBooks**.
2. In the **Rulesets** tab, click the **VScan** rule set.
3. Click the **padlock** icon.
4. Expand the **VScan** rule set.
5. Set the directory and the number of image files to process:
 - a. Set the image file to 1.

The SetMaxImageFiles() action is used here. Setting the value to 1 gives you one image for each batch. If documents are coming from a fax, set the value to 1. If the documents are scanned from a physical scanner, batches of 50 or 100 image files is preferred. The setting depends on the source where the documents come from.

- b. Add the VScan SetMultiPageTiff from the Actions library. The scan action must be last in the rule set.
 - c. Click **Save**.
6. Click **Publish** to save the rule set.

7.2 Adding a document with pages and fields

Now you can add a document to an existing project. This task includes adding the potential page types that this document might have and the fields that you might want to capture for each page type if applicable.

For the example in this book, we create a document, called *Estimate_Invoice*, with the following page type and field information:

- ▶ Document name: Estimate_Invoice
- ▶ Potential page types:
 - Main_Page
 - Trailing_Page
- ▶ Potential fields to be captured in the Main_Page
 - Doc_Title
 - Pol_Number
 - Client_Number

- Van_Number
- Van_Name
- Ref_ID
- Ref_Date
- Ref_Total

► Potential fields to be captured in the Trailing_Page: None

To add the new document to the existing project, complete these steps:

1. Open Datacap Studio.
2. Click the **Lock DCO for Editing** icon.
3. Right-click **Batch Level**, and select **Add** → **Document**.
A new document, labeled *Document 1*, is added.
4. Rename the document to Estimate_Invoice.
5. Add the potential pages to this document, which contains two potential types of pages:
 - a. Right-click the document, and then select **Add Multiples**.
 - b. Select **Pages**, and then enter 2.
Two new pages, labeled *Page1* and *Page2*, are added.
 - c. Rename Page1 to Main_Page, and then rename Page2 to Trailing_Page.
6. Add fields that can be captured by the page. Complete the following steps for Main_Page:
 - a. Right-click **Main_Page**, and then select **Add Multiple**.
 - b. Select **Fields**, and enter 8, which specifies that potentially eight fields must be captured for this page type.
 - c. Rename the fields to Doc_Title, Pol_Number, Client_Number, Van_Number, Van_Name, Ref_ID, Ref_Date, and Ref_Total (Figure 7-1 on page 256).
7. Click **Save**.
8. Click the **Unlock DCO for Editing** icon.

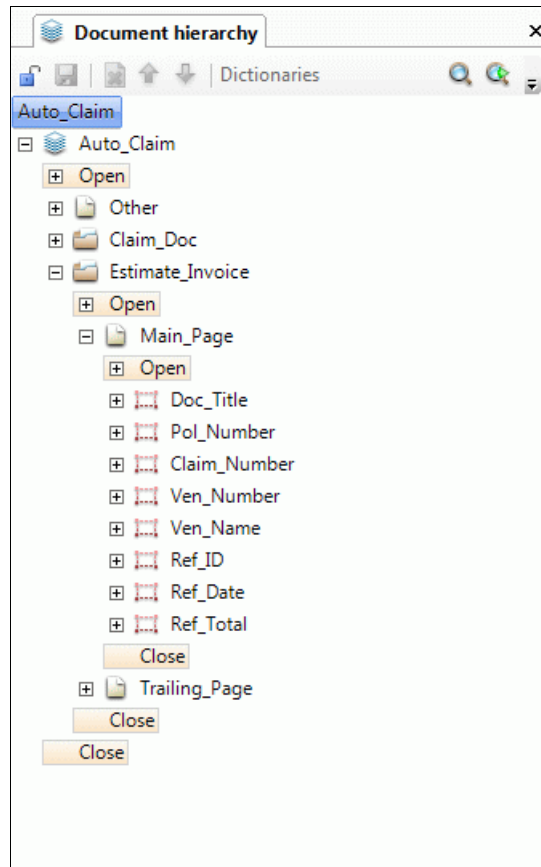


Figure 7-1 Document hierarchy tab

7.3 Setting the PageID logic

In Datacap Studio, set the PageID logic:

1. On the **Rulesets** tab, expand the **PageID** rule set for PIE_RBooks.
2. Add the `rrCompare()` action to Page from Fax Job in the PageID rule set.
3. Configure the `rrCompare()` action to check if the value FAX is found at the variable level job type:
 - a. Highlight the **rrCompare()** action that you just added.
 - b. On the **Properties** tab, for string object1, enter FAX. Then for string object2, enter `@B.JobType`.

4. Add an action, SetDCOType(), that will be triggered if the rrCompare() action is *true*:
 - a. Add the **SetDCOType()** action to the Page from Fax Job.
 - b. On the **Properties** tab, for the StrParam string, enter Main_Page.
5. Add a function, Function 1, to the PageID rule set:
 - a. Right-click **PageID**, and then select **Add New Function**. Function 1 is added to the bottom of the list.
 - b. Click the **up arrow** to move the new Function 1 to the top.
 - c. Add the ChkLastDCOType() action from the **Actions library** tab to Function1.
 - d. On the **Properties** tab, for the StrParam string, enter Main_Page.
 - e. Add the SetDCOType() action from **Actions library** tab to Function1. This action runs after the ChkLastDCOType() action.
 - f. On the **Properties** tab, for the StrParam string, enter Trailing_Page, which sets the current page type to Trailing_Page.
6. Add another function, Function 2, to the PageID rule set:
 - a. Right-click **PageID**, and select **Add New Function**. Function 2 is added to the bottom of the list.
 - b. Click the **up arrow** to move the new Function 2 under Function1.
 - c. Copy and paste the actions from Function1 to Function 2.
 - d. Change the parameter for the newly copied ChkLastDCOType() action in Function 2 to Trailing_page.
7. Rename Function 1 to Looks for Fax Trailing.
8. Rename Function 2 to Looks for additional Fax Trailing.
9. Save and publish this rule set.

The functions added to the PageID are triggered sequentially. For example, a page comes in from a fax machine. Initially, the page type is *Other*, causing the first two functions, Looks for Fax Trailing and Looks for additional Fax Trailing, to fail. The Page from Fax Job function is triggered because the page came from the fax. It sets the page type to Main_Page. If additional pages come from the fax immediately following the first page, the Looks for Fax Trailing function checks whether the previous page was the main fax page (with the page type of Main_Page).

If the previous page is of the Main_Page page type, it sets the page type to *Trailing_Page*. If additional pages proceed, the Looks for additional Fax Trailing function is triggered because the last page had a Trailing_Page type. The

additional pages are all set to the `Trailing_Page` page type. Every additional page that comes in from this fax action is labeled as a `Trailing_Page`, based on the Looks for additional Fax Trailing function.

If a page comes in that is from a scan instead of a fax, the Looks for Fax Trailing, Looks for additional Fax Trailing, and Page from Fax Job functions fail. They trigger any other functions that we might have set as part of this rule set, based on the source and functions we might have.

7.4 Adding the Image Enhance feature to the pages

After creating page types for your document, you can configure Datacap to clean up the images of these page types by adding the Image Enhance feature to them.

On the **Rulesets** tab, look at the `Enhance_Image` settings in ImageFix. These settings are set at the batch level. We must add this rule to the newly created page types, which are the `Main_Page` and `Trailing_Page` pages in this example.

To add the Image Enhance feature to the page types, complete these steps:

1. On the **Document hierarchy** tab, click the **Lock DCO for Editing** icon.
2. Navigate to the object to which you are changing, which is `Main_Page` in this example.
3. Verify that the `Enhance_Image` rule set is highlighted, and then click **Add to DCO**.
4. Repeat steps 2 and 3 for the `Trailing_Page` page type.

Now when pages of either the `Main_Page` or `Trailing_Page` type are triggered, the imaging enhancement runs as part of it.

7.5 Configuring the CreateDocs rule set for the pages

Next we configure the CreateDocs rule set. For this rule set, we do not need to change any of its components. The CreateDocs rule set contains two rules: the Create Docs rule and the Create Fields rule.

The Create Docs rule is part of the batch-level object. This rule contains a create document action, which examines all the pages in the batch. When it detects a `Claim_Pg` page, for example, it sets the start of a new document.

The Create Fields rule set triggers at each page that has fields attached. The rule set creates the fields in the Claim_Pg page.

We add the Create Field rule to the Main_Page under the Estimate_Invoice document by using the following steps:

1. On the **Document hierarchy** tab, highlight the **Main_Page** object.
2. On the **Rulesets** tab, highlight the rule **Create Field**.
3. Click **Add to DCO** to add the rule to the page.
4. Right-click **Main_Page**, and then select **Manage Variables**.
5. Under Object General information, complete these steps:
 - a. Change Max value to 1.
 - b. Change Min value to 1.
 - c. Change Order to 1.
 - d. Click **Done**.

These settings dictate that every Estimate_Invoice has a minimum of one main page (Main_Page page type), a maximum of one main page, and no more than one main page for each document.

For the trailing pages (pages of type Trailing_Page), we keep the default settings to 0 for Min, Max, and Order because we can have multiple trailing pages for each document or we might not have any trailing pages.

6. Save and unlock the Document Hierarchy (DCO).

7.6 Adding a full page OCR rule set to the pages

We have now completed the setup for the new Main_Page document with the fields and the page structures. We have added the PageID, ImageFix, and the CreateDocs rule sets. Now we add the pages into a document structure and start the recognition process (recognize action).

For the recognize action, we have the Recognize Page rule set, which has the RecognizePageOCR() action. This rule can be used for the Main_Page that we created.

To add a full page Optical Character Recognition (OCR) rule set to the pages, complete these steps:

1. Lock the **Document hierarchy** tab for editing.
2. Select the **Main_Page** page.

3. On the **Rulesets** tab, select the **RecognizePageOCR()** rule, and then click **Add to DCO**.
4. Select **Trailing_Page**.
5. Select the **RecognizePageOCR()** rule, and then click **Add to DCO**.

Now when this rule set is run, it does a full page OCR on all the pages and create a CCO for each of these. Then we can use the CCO to extract data out of it.

7.7 Setting the fingerprint for the Estimate_Invoice document

The next rule set is the fingerprint rule set, which was previously set. This rule includes Batch Level Fingerprint Settings, which are tied to the batch level. This rule set contains the following basic actions:

- ▶ **SetFingerprintsDir()**, which identifies where the fingerprints are stored
- ▶ **SetProblemValue()**, which indicates the level of match that we need to have a valid match
- ▶ **SetFingerprintSearchArea()**, which indicates the percentage of the page we want to look at

The second rule, Claim Page Fingerprint Matching, is attached to the claim page (Claim_Pg). In this rule, the FindFingerprint() action is set to *true*, which we switch to *false*. Because this rule is on the claim page, we do not want the system to create fingerprints by default.

This rule was originally used as part of the claim page, which is a fixed form that we used while setting the main system up for the first time. Now that the system is set up, we change the FindFingerprint() action to *false*.

For the Estimate_Invoice document, which is a form coming from an outside source, we do not have control of the form. This form is also a live form, in the sense that it might be constantly changing. For the Estimate_Invoice document, we set FindFingerprint() to *true*. In return, if this form is new, it allows the system to learn it instantly.

To add the FindFingerprint() rule set, complete these steps:

1. Verify that the **Document hierarchy** tab is locked for editing.
2. On the **Rulesets** tab, lock the Find Fingerprint rule set.
3. Right-click the **Claim Page Fingerprint Matching** rule, and then select **Copy**.

4. Right-click the **Find Fingerprint** rule set, and then select **Paste** to add a copy of the rule.
5. Rename the new rule to **Main Page Fingerprinting**. This rule is attached to the main page.
6. Highlight the **Main_Page**, and click **Add to DCO**.
7. Select the **FindFingerprint()** action from the Claim Page Fingerprint Matching rule.
8. On the **Properties** tab, in the parameter section, change the value to **False**.
9. Save changes and publish the rule set to update the rule set with the new information.

7.8 Obtaining field values in the Claim_Pg document by using the Locate() rule set

The Locate() rule set is used to find the data on the page. On the Claim_Pg, we were used a generic action that worked with known, fixed-field locations. The Locate() method we used searches for data that is in a fixed location. It also searches for data that might be in different places on the page.

We create a separate rule for every data field on the main page:

- ▶ Doc Title field
- ▶ Pol_Number field
- ▶ Claim_Number field
- ▶ Vendor Number field
- ▶ Vendor Name field
- ▶ Ref ID field
- ▶ Ref Date field
- ▶ Ref Total field

7.8.1 Doc Title field

The Document Title field is a drop-down list box that contains the choices of Invoice or Estimate. The document title is used to identify the type of forms we process. To populate the field, use a regular expression search for the word “Estimate.” If the word is found, we set the field value to Estimate. Otherwise, the default is set to Invoice.

7.8.2 Pol_Number field

The Pol_Number field contains the policy number. This field is special in that it is not populated through OCR or Intelligent Character Recognition (ICR). This field is populated during the Verification phase.

7.8.3 Claim_Number field

The zonal recognition method tries to match the Claim_Number field with a known location from a known fingerprint by using the PopulateZNField() action. If the current image matches with a known fingerprint, it goes to that zone and then extracts the value.

7.8.4 Vendor Number field

To populate the Vendor Number field, we match against a known fingerprint. That value is then used to look for a corresponding value in a database. If this time is the first time the form is seen, or the form does not match any known fingerprint, this field is blank and requires manual interaction during the Verify process. The Vendor Number field requires a valid Vendor Name field value.

7.8.5 Vendor Name field

To populate the Vendor Name field, we match against a known fingerprint. That value is then used to look for a corresponding value in a database. If this time is the first time the form is seen, or the form does not match any known fingerprint, then this field is blank and requires manual interaction during the Verify process.

7.8.6 Ref ID field

The zonal recognition method tries to match this field with a known location from a known fingerprint by using the PopulateZNField() action. If the current image matches a known fingerprint, it goes to that zone and then extracts the value.

7.8.7 Ref Date field

The zonal recognition method tries to match this field with a known location from a known fingerprint by using the PopulateZNField() action. If the current image matches a known fingerprint, it goes to that zone and then extracts the value.

7.8.8 Ref Total field

The Ref Total field has a unique aspect to it in that, although it is always part of the document, its position on the document can vary from form to form. On some forms, the total might not be displayed on the first page. To handle this issue, we incorporate two new features into our project, the MergeCCOs_byType() and FindLastKeyList() actions.

We add the MergeCCOs_byType() action at the document level. This action takes the CCO data from the main page (Main_Page) with all of the trailing pages (Trailing_Page) within the document, making one multipage CCO.

We add the FindLastKeyList() action at the field level. This action calls a text file ending with the .key extension. Then it starts with the first word in the list and the bottom of this multipage CCO file to look for a match. If the first word does not match, it uses the second word, then the third word, and so on.

7.9 Looking up vendor information from a database by using the Lookup() rule set

The Lookup rule set retrieves data values from the database. For this example, we want to look up the vendor name and vendor number.

7.9.1 Looking up a vendor name

After a successful fingerprint match, we now know the template ID for the matching fingerprint. To look up the vendor name, we create a rule set that uses the following rules:

1. At the batch open, the first rule establishes the connection to the fingerprint database.
2. Set on the field, the second rule uses Main_Page.TemplateID to get the matching fingerprint class:

```
"SELECT Host.hs_RefName FROM Host INNER JOIN Template ON  
Host.hs_HostID = Template.tp_HostID WHERE (((Template.tp_TemplateID)  
LIKE '%s'));"
```

This rule looks up the template ID from the fingerprint database. Then, based on the template ID, it returns the matching hs_RefName (fingerprint class) that the template is stored under. If a new fingerprint was created during the Find Fingerprint rule set, this value is <New>.

3. At the batch close level, the third rule closes the connection.

Placing these rules this way allows for a more efficient use of the processing sequence.

7.9.2 Looking up a vendor number

The vendor number lookup is similar to the Lookup rule set. The main differences are the database to which it connects and the SQL statement that it executes. With this rule set, we want to achieve the following goals:

- ▶ Validate the vendor name with our known acceptable vendors
- ▶ Retrieve our internal number that we assigned to this particular vendor

Like the Lookup rule set, we have the following rules:

1. At the batch open level to establish the connection.
2. At the Vendor Number field to execute the SQL statement to retrieve the number based on the Vendor Name field.
3. At the batch close level to close the connection.

The actual SQL statement might vary based upon the database to which you connect.

7.10 Validating data

Data validation is when you determine whether the data you have captured meets the rules for data integrity as defined in the business requirements. For example, when we established the business rules for the application, we decided to test whether the cost fields are in a valid currency format. A validation failure does not mean that the original page contains invalid data. It might mean that the recognition engine failed to recognize one or more characters correctly. Whatever the reason for the error, you can set the page status to make sure that the page is displayed to an operator for verification.

This section highlights the validation techniques that we use in our use case. We validate against the following objects:

- ▶ Page Level field
- ▶ Doc Title field
- ▶ Pol_Number field
- ▶ Vendor Number field
- ▶ Vendor Name field
- ▶ Ref ID field
- ▶ Ref Date field

7.10.1 Page Level field

At the page level, we add a rule that contains the `StatusPreserveOFF()` action. With this action, the page status can be controlled by the fields. When a field fails its validation rules, the field status is set to 1. When this failure happens, it causes the page status to also be set to 1.

7.10.2 Doc Title field

By default, the Doc Title field is set to Invoice, unless during the Locate phase, the document is determined to be an Estimate_Invoice. No validation is available for this field.

7.10.3 Pol_Number field

The Pol_Number field has a few basic validation rules that we need to follow. First, we use the `allowOnlyChars()` action with the `1234567890CL-` parameter. This parameter removes any character that is not in the list. Next we check the minimum length and the maximum length by using two actions for our use case: `IsFieldLengthMin()` with a parameter of 4 and `IsFieldLengthMax()` with a parameter of 10.

7.10.4 Vendor Number field

The validation for the Vendor Number field involves checking to see if it is populated. If it is populated, it does a database lookup to see if this value and the value from the Vendor Name field matches in the database.

7.10.5 Vendor Name field

The validation for the Vendor Name field involves checking to see if it is populated. It also entails doing a database lookup to see if this value and the value from the Vendor Number field match in the database.

7.10.6 Ref ID field

Because the value of this field is controlled by the business that is creating the estimate and invoice forms, our validations here are limited. We can use an `IsFieldLengthMin()` action with a parameter of 3 to ensure that the field is populated. We can also use the `IsFieldPercentNumeric()` action and set the

parameter to 60. This setting helps to ensure that the field contains at least 60% numeric characters.

7.10.7 Ref Date field

The Ref Date field is the date on the estimate or the invoice form. This date must be in a MM/DD/YYYY format. We can enter this date by using the `IsDateWithReformat()` action and the MM/DD/YYYY parameter. We can also check to ensure that the date is recent, but not a date in the future, by using the `IsFieldDateWithinXRange()` action and setting the parameter to 90. If the date is in another format, the action normalizes the date into this format.

7.11 Routing scanned pages

After the unattended validation runs, the Routing rule set helps to determine which pages will be presented to the Verify operator for manual validation. Within the Validate rule set, every page had a status of 0 or 1 assigned to it, where a 0 means it passes all field level validations, and a 1 means it failed on one or more field validations.

The Routing rule set checks the page status value. If the page status is set to 1, no more additional checks are done because this page is already marked for operator intervention. When the page status is to 0, the Routing rule applies. The Routing rule then runs an action that examines the character confidences of every character on the page. If it finds any confidence values that are less than 8, it sets the page status to 1, marking this page as *verified*.

7.12 Verify task with Batch Pilot

During verification, you display pages to an operator for manual checking and possibly correction. You display pages to an operator for two reasons:

- ▶ A page contains one or more characters or OMR fields that were marked “low confidence” by the recognition engine.
- ▶ A validation rule failed, indicating a problem with the integrity of the data.

This section covers the verification user interface, Batch Pilot. The rules that are used here are described in 7.10, “Validating data” on page 264. We want to reuse the same rule to ensure that the reason that the page went to the Verify operator has been addressed. We also want to ensure that the operator does not introduce a new error while working on the page.

7.12.1 Creating a Verify panel

To create a Verify panel for the Estimate and Invoice forms, complete these steps:

1. Select **Start** → **All Programs** → **Datacap** → **Batch Pilot** → **Batch Pilot** to launch the Batch Pilot program.
2. Select **File** → **Open Project**.
3. Select the **rrs_verify.bpp** project, and then click **Open**.
4. In the bottom section of the Batch Pilot window, expand **UPDATE** → **UPDATE WITH DOCUMENT**.
5. Right-click the **UPDATE WITH PAGE** page, and select **AutoForm**. AutoForm reads the document hierarchy (setup DCO), and an image snippet control and an edit or list box control are displayed for each of the defined fields.
6. Select **File** → **Save Form**.
7. Open the **UPDATE\verify** folder, and then save the form as **Estimate.dcf**.
8. In the bottom section of the Batch Pilot window, right-click the **UPDATE** page, and select **Pick form**.
9. From the **UPDATE\verify** folder, select **Estimate.dcf**, and click **Open** to link the form to the page type.
10. Click **File** → **Save Project**.

7.12.2 Determining which pages to display

By default, Batch Pilot displays all pages to the operator, regardless of whether a given page has a problem. In this section, we configure Batch Pilot to display only pages with status of 1, which indicates a problem.

To display only pages with a status of 1, complete these steps:

1. In the Taskmaster Client window, click **Administrator**
2. On the **Workflow** tab, expand **Main Job** → **Verify** and then select **Setup**.
3. In the Batch Pilot window, click **File** → **Task Settings**.
4. Click the **Filters** tab.
5. Under Type, select **Main_Page**. Then select **Level: PAGE, Property: STATUS, Problem Value: 1** and click **Add**. Now if we are in the Verify panel and the page has a status of 1, indicating a problem, then the page is displayed to an operator. Click **OK**.
6. Click **File** → **Quit**.

7. Click **Yes** to save the project.
8. Click **Apply**, and then click **Done** to close the Administrator window.

7.13 Export task

Taskmaster can export data to a text file, an XML file, a database, a document management system, or a custom business process. The default output format is a text file. This section looks briefly at the actions that you can use to export data to a document management system, IBM FileNet P8, and a database. For the document management system, we have a use-case requirement that all of the estimates and invoices are sent to FileNet P8 as text searchable PDF/A documents.

Making a PDF/A per Estimate and Invoice document

Now that we have extracted the data from the form and verified that it is correct, we can move to the next task, Export. The use-case scenario requires that we create a single PDF/A file for each estimate or invoice document regardless of how many pages are in the document. Because the pages are now broken into documents, we can run a rule at the document level to create the PDF:

1. In Datacap studio, on the **Rulemanager** tab, on the **Rulesets** subtab, right-click **Auto_Claim**, and then select **Add Ruleset**. This step adds a rule set, called *Ruleset1*, to the bottom of the list.
2. Rename Ruleset1 to Estimate Invoice PDF creation.
3. Select **Function1**. From the OCR_S actions library, add the action RecognizeDocToPDF() to it.

The RecognizeDocToPDF() action has five available parameters, with a numeric value indicating the Document Format type:

- 1 A PDF with an image on text
- 2 A PDF where Graphics, KeepBold, KeepItalic, and KeepUnderline have an effect
- 3 A PDF with image substitutes where Graphics, KeepBold, KeepItalic, and keepUnderline have an effect only
- 4 A PDF with image on text where Graphics, KeepBold, KeepItalic, and KeepUnderline have an effect only
- 5 A PDF image only.

We choose the first option. These options come directly from the help file of the action.



Implementing the business process component

A key component of IBM Production Imaging Edition is the business process management component that is provided by IBM FileNet Business Process Manager (BPM) Tools. This chapter describes how to implement the Auto Claim business process of the solution example that is introduced in Chapter 4, “Solution example” on page 123.

This chapter includes the following sections:

- ▶ FileNet Business Process Manager Tools
- ▶ Running the Auto Claims process with the FileNet BPM Tools
- ▶ How the Auto Claim process works
- ▶ Business Process Manager step configuration

8.1 FileNet Business Process Manager Tools

IBM Production Imaging Edition provides the IBM FileNet BPM Tools. These tools come with a Process Engine and a set of associated components and applications so that you can create, modify, manage, simulate, and analyze your business processes.

IBM FileNet BPM Tools can work with active content, such as loan applications, claim applications, email messages, or faxes. By working with this content, the tools can trigger an automated process within IBM FileNet P8. As a result, scanned images from IBM Datacap Taskmaster Capture (Taskmaster) can be stored in the IBM FileNet P8 repository and invoke a business process associated with their document class.

In order for a process to be defined, a process definition is created by using the Process Designer. Process Designer is a visual tool that is used to map the activities and resources required to accomplish a business process. These activities or steps (nodes) are connected by routes (arcs), which define the sequence in which the steps are executed.

For information about using and building business process with IBM FileNet BPM, see the IBM Redbooks publication *Introducing IBM FileNet Business Process Manager*, SG24-7509.

For your convenience, the following sections provide a brief review of the key concepts and building blocks of a business process.

8.1.1 Steps

The steps in a process represent specific business or system activities. Activities can be performed by a user, a group of users, or an automated application.

Several types of steps are available:

Launch step	The first step in a process. Every process has this step.
General step	Can be associated with a user, or a group of users, who must process the work item to complete the step. It can also be sent to a work queue instead of a defined user or group.
System step	Is performed by the system, such as assigning data field values, creating a process, or suspending a process for a period of time or until a condition occurs.
Submap step	Calls another map in the current process definition.

Component step	Calls an external application or system by using IBM Component Integrator.
Web services step	Invokes or implements a web service to allow integration with external applications and services.

8.1.2 Routes

Routes define the order in which the execution occurs for a series of steps. Routing is achieved by defining rules against data fields and user responses, such as Approve and Deny. If a boolean expression is defined on a route, the route is traversed only if the expression evaluates to *True*.

8.1.3 Maps

A Workflow collection contains workflow definitions, which contain maps. Maps contain steps, routes, and text annotations.

A map represents the sequence of steps and routes that is required to complete a process. You can define maps by using the Process Designer or import them from other process design tools such as Microsoft Visio. (Additional licenses might be required.)

8.1.4 Content events

When an object is created, edited, or deleted in a repository, a process can be automatically launched. The properties and data associated with the object can be used to determine the routes taken in the business process.

8.2 Running the Auto Claims process with the FileNet BPM Tools

The use case defined in this Redbooks publication details a scenario where a client initiates an insurance claim for a car accident. This section explains how to map this business process by using the Process Designer (a FileNet BPM Tool) component of Production Imaging Edition. It also takes you through the configuration process.

Figure 8-1 shows the process developed for the Auto Claim use case in the IBM Process Designer. The process is created by dragging steps from the step palette to the Process Designer window. The steps are then connected by using routes to define the sequence in which the process executes.

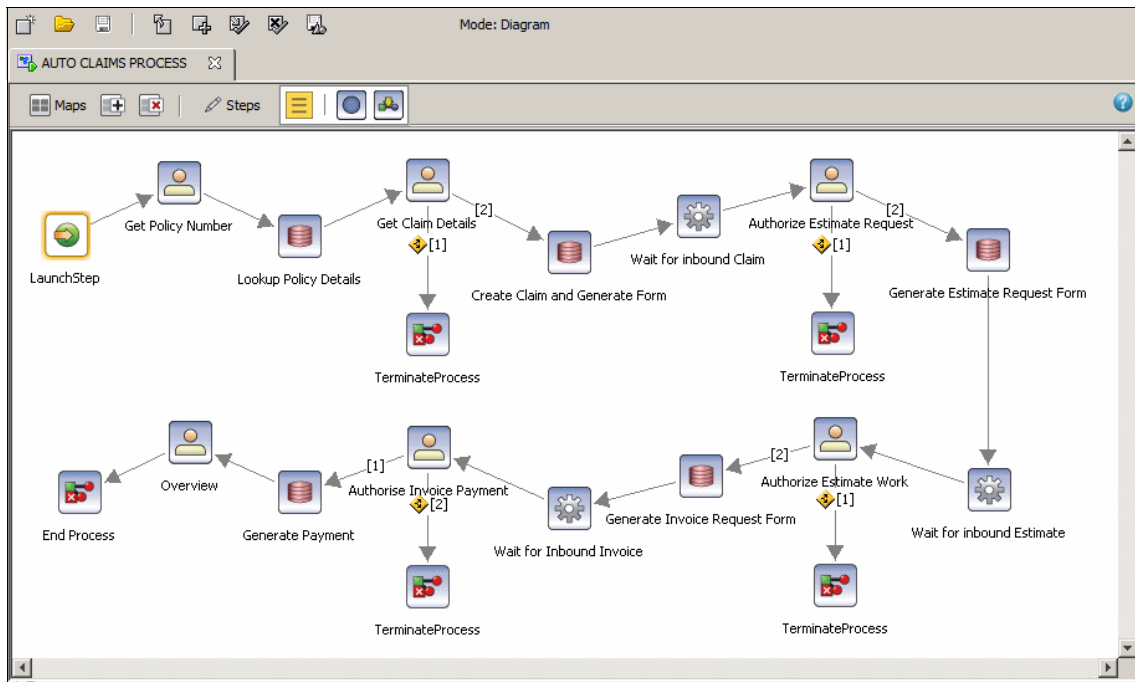


Figure 8-1 Business process in Process Designer

Data fields required in the process were defined within the Process Designer and exposed at varying steps of the process as needed. Conditions were also applied to the routes to define which direction the process should flow.

In this process, we use the process steps shown in Figure 8-2 on page 273:

Launch

The start of the process. In this step, we can define any required fields and their value.

WaitForCondition

A system step. This step can be configured to allow the business process to wait for a condition event in another process to trigger it to continue.

Approval

With this step, a defined user or group can grant or deny approval at a given stage of the process. This step can also involve a user adding additional details or comments to specified fields (providing the user has the appropriate privileges).

- DBExecute

With this step, the process can call a stored procedure in a predefined supported database. Use of stored procedures over SQL hides the complexity of the database being queried and instead requires only the input and output fields to be defined.
- Terminate

Terminates the workflow. In the process that we developed for the Auto Claim use case, we used the Terminate process steps for the end of the process and when the claim was not authorized at given approval points. Alternatively, we might have used submaps to invoke additional workflows. However, to keep the process simple, we decided to terminate the process.

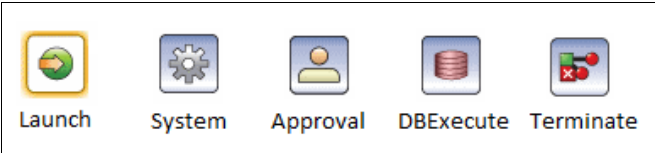


Figure 8-2 Process steps

By using these steps, we built a business process by using FileNet BPM Tools for FileNet P8 that automates the Auto Claim use case.

8.3 How the Auto Claim process works

The client calls the insurance call center and speaks with an agent about the client’s claim. The agent launches the business process in FileNet P8 and asks the client for the policy number. Figure 8-3 illustrates the “Get Policy Number” step.

Items Found: 1

	Name ▲	Step Name	Status	Received On
	Auto Claim Process	Get Policy Number	In Progress	24/06/11 16:21

Figure 8-3 Launching the Get Policy Number user interface

Figure 8-4 illustrates where the agent enters the policy number of the client.

Properties:

POLICY_NUMBER:

43398313

Figure 8-4 Entering the policy number

The policy number is a unique identifier (primary key) that is submitted to the system. With this number, the system can locate and retrieve the clients policy details by using a database stored procedure.

The agent requests information about the claim that the client wants to make. This information is entered into the system (Figure 8-5). A new claim is created with this information and associated with a new unique claim number. The claim number is automatically generated by the database to ensure its uniqueness.

Properties:

CLAIM_DATE:	03/02/11
CLAIM_TIME:	13:30
DESCRIPTION:	Incident occurred on Highway 42
GENERATE_CLAIM_FORM:	True
POLICY HOLDER ADDRESS:	"1223 SPRINGFIELD DRIVE, NY"
POLICY HOLDER NAME:	"JOHN DOE"
POLICY HOLDER ZIP:	"234567"
POLICY_NUMBER:	"43398313"
VEHICLE_COLOR:	"Silver"
VEHICLE_LICENSE:	"5566TY"
VEHICLE_MANUFACTURER:	"Lexa"
VEHICLE_MODEL:	"Charger"
VEHICLE_VIN:	"2B3AA4CTX82KP3456"

Figure 8-5 Retrieved policy details and claim details entered

A background task (not shown in the process) queries the database for new claims that require claim form generation. A claim form is generated and sent to the client by using a post for completion process.

This main process waits for a condition to occur (Figure 8-6). The condition is for an inbound claim form workflow to have the same claim value that the main process is waiting for. When this condition occurs, the workflow effectively closes the loop and indicates that the form has been returned and processed correctly.

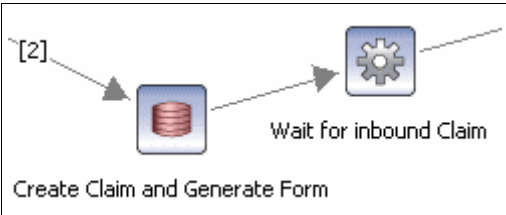


Figure 8-6 Stored procedure and a System WaitForCondition

The client receives the form by mail and then completes the form by selecting a check box, signing the claim, and entering a date on the form. The client then mails the form back to the given address of the insurance broker.

The mailed form is scanned by using a thick client at the insurance broker (Figure 8-7). Taskmaster identifies the relevant data, applies OCR technology, and then extracts the relevant data from the form.

Auto Insurance Claim Form
Insurance Company A

Policy Holder Address: John Doe
123 Springfield Drive
NY 12345

Policy Number: 4339013
Incident Number: CL-41
Incident Date: 03/06/11
Incident Time: 12:30

Vehicle License: 5000TY
Vehicle Colour: Silver
Vehicle Manufacturer: Lexus
Vehicle Model: Charger
Year of Manufacture: 2008
Chassis Number (VIN): 203AAAC780NKP0000

Incident Description:
Incident occurred on Highway 401. 901.
Damage occurred to front bumper and front right headlight.
Airbag was triggered.

Was anyone injured in this incident which required medical attention? ☐ Yes ☒ No

By signing the adjacent box and entering the current date, you agree the information above is accurate to the best of your knowledge.

Signature: [Signature]
Date: MM/DD/YYYY
03/06/11

<< < > >>|

Image: 1 of 1 Doc: 0 of 0 Go

Figure 8-7 Scanned claim form in Taskmaster

Any verification of data can be carried out in the verify process if required (Figure 8-8).

IncidentNumber	CL-47	CL-47
PolicyNumber	43398313	43398313
PolicyEffectiveDate		10/10/2010
PolicyExpiryDate		10/10/2011
InsuredName	John Doe	JOHN DOE
ChasisNumber_VIN	2B3AA4CTX82KP3456	2B3AA4CTX82KP34
Injury	<input type="checkbox"/> Yes <input checked="" type="checkbox"/> No	No
Veh_Make	Lexa	Lexa
Veh_Model	Charger	Charger

Figure 8-8 Verify window in Taskmaster


After the Verify task is completed, Taskmaster commits the claim form to IBM FileNet P8 as a specifically defined document class. The document class has an associated subscription that is triggered as soon as the document is committed to FileNet P8.



The subscription starts the associated inbound workflow, which extracts the properties of the document object (fields from Datacap) and writes them into workflow fields (Figure 8-9). In turn, this action meets the waiting condition of the main workflow and allows it to continue. The main workflow also copies the location of the scanned document from the inbound triggered workflow, making it available in the main flow.

Properties:

CLAIM_DATE:	"03/02/11"
CLAIM_NUMBER:	"47"
CLAIM_TIME:	"13:30"
DATED:	"030611"
DESCRIPTION:	"Incident occurred on Highway 42, NY. Damage occurred to front bumper and front right headlight. Airbag was triggered"
GENERATE_ESTIMATE_REQUEST:	<input type="checkbox"/> True
INJURED:	"N"
POLICY_EFFECTIVE_DATE:	"10/10/2011"
POLICY_EXPIRY_DATE:	"10/10/2010"
POLICY HOLDER ADDRESS:	"1223 SPRINGFIELD DRIVE, NY"
POLICY HOLDER NAME:	"JOHN DOE"
POLICY HOLDER ZIP:	"234567"
POLICY_NUMBER:	"43398313"
SIGNED:	"Y"

Attachments:

 **CLAIM_DOCUMENT** Items Found: 1

 **Claim Form** 

Written by p8admin and last modified by p8admin on 24/06/11 16:25

Figure 8-9 Display of scanned fields within FileNet P8

The inbound invoice workflow now completes and ceases to exist. The main workflow continues to the next step.

A user confirms that the claim form is correctly completed and requests an estimate to be granted. Rules have already been applied within Datacap to ensure that all fields are populated, which is merely a manual check and might not be required.

The workflow attachment is made available for manual inspection by the agent.

Figure 8-10 shows a claim form.

Auto Insurance Claim Form
Insurance Company A

CLAIM 25A1

Policy Holder Address : John Doe,
123 Springfield Drive
NY 234567

Driver Name : John Doe

Policy Number : 43398313

Incident Number : CL-47

Incident Date: 03/02/11

Incident Time: 13:30

Vehicle License : 5666TY

Vehicle Colour : Silver

Vehicle Manufacturer : Lexa

Vehicle Model : Charger

Year of Manufacture : 2008

Chassis Number (VIN) : 2B3AA4CTX82KP3456

Incident Description:

Incident occurred on Highway 42, NY.
Damage occurred to front bumper and front right headlight.
Airbag was triggered.

Was anyone injured in this incident which required medical attention? ☐ Yes ☒ No

By signing the adjacent box and entering the current date, you agree the information above is accurate to the best of your knowledge.

Signature: John Doe

Date: 03/06/11

Figure 8-10 Sample claim form

Upon acceptance of the claim form, a stored procedure is now invoked to update the status of the claim, indicating that an estimate request form is sent to the client. Data captured from the form is also committed to the back-end database.

The process now goes into another wait condition, awaiting an estimate response from a work shop for repair work (Figure 8-11). This condition is identified by the claim number of the claim.

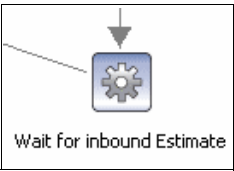


Figure 8-11 WaitForCondition step

The client receives the estimate request form and requests an estimate from the repair shop. This request is created by the repair shop and faxed back to the insurance brokers head office. The form includes the claim number of the form as a reference.

The fax server receives the estimate form. Then Taskmaster ingests and identifies the relevant data, applies OCR technology, and processes the document. Then Taskmaster finally commits the information to FileNet P8 (Figure 8-12).

Properties:

CLAIM_NUMBER:	"47"
DESCRIPTION:	"Incident occurred on Highway 42, NY. Damage occurred to front bumper and front right headlight. Airbag was triggered"
ESTIMATE_DATE:	"03/12/11"
ESTIMATE_REPAIR_SHOP_ID:	"33675"
ESTIMATE_REPAIR_SHOP_REFERENCE:	"VBDA12333"
ESTIMATE_TOTAL:	"475.69"
GENERATE_INVOICE_REQUEST:	<input type="checkbox"/> True

Attachments:



	ESTIMATE_DOCUMENT
	CLAIM_DOCUMENT

Figure 8-12 Claim information displayed in FileNet P8

Similar to the claim form, the estimate form has a document class associated to it that has a workflow subscription. The workflow is triggered, and the relevant fields are mapped from the document to the workflow fields. The workflow claim

number satisfies the WaitForCondition of the main workflow and allows it to continue its process.

The main process can now continue. It informs an agent that the estimate form has been received. It is made available for view in the user interface.

The agent now determines if the estimate is suitable for claim. The agent has access to both the claim and estimate documents if any inspection of the documents is needed.

At this point, the agent can agree to proceed with the work or investigate further. For this use case, we assume that the agent will continue. However, another process can be invoked to deal with this claim and process it accordingly.

Assuming that the agent wants to continue to authorize the work, another stored procedure is invoked to set a flag in the database for a back-end system to generate an invoice request. It also stores the values held for the estimate and associated repair shop. The process now waits for the invoice to be received in relationship to this claim.

The back-end process generates a request for invoice form, which is mailed to the client.

The client receives this letter and requests that the work, according to the estimate, is carried out. The work is completed. Then the invoice is created by the repair shop and sent by fax back to the head office of the insurance broker. The form includes the claim number.

The fax server receives the invoice form. Then Taskmaster ingests and identifies the relevant data, applies OCR technology, and processes the document. Then Taskmaster finally commits the information to FileNet P8.


Similar to the estimate form, the invoice form has a document class associated with it that has a workflow subscription. The workflow is triggered, and the relevant fields are mapped from the document to the workflow fields. This workflow claim number satisfies the WaitForCondition of the main workflow and allows it to continue its process.


The main process can continue. It now informs an agent that the invoice form has been received. It is made available for view in the user interface (Figure 8-13).

Properties:

DESCRIPTION:	"Incident occurred on Highway 42, NY. Damage occurred to front bumper and front right headlight. Airbag was triggered"
ESTIMATE_REPAIR_SHOP_ID:	"33675"
ESTIMATE_TOTAL:	"475.69"
INVOICE_REPAIR_SHOP_ID:	"33675"
INVOICE_TOTAL:	"475.69"

Attachments:

 CLAIM_DOCUMENT

 ESTIMATE_DOCUMENT


 INVOICE_DOCUMENT

Figure 8-13 Claim, Estimate, and Invoice document attachments and claim information

The agent now determines if the invoice is suitable for claim. The agent has access to the claim, estimate, and invoice documents if any visual inspection is needed.

This step can be further automated if required. For example, the value for the estimate and the invoice do not match, or they are not within a certain tolerance of each other. Only then must an agent manually inspect the data and confirm progress of the process.

If the agent confirms the process can continue, a stored procedure is called to update the back-end system to authorize payment to the repair shop and to insert the value of the invoice captured along with repair shop details.

Finally, the user is presented with all the data held in the flow (Figure 8-14 on page 282). This step was created to show all data captured within the life of the business process.

p8admin | Friday, 24 June 2011 Help

Views

Task

[Milestones](#)

Actions

Reassign

Track Status

Overview

Deadline: None specified

Subject: Auto Claim Process

Properties:

CLAIM_DATE:	03/02/11
CLAIM_NUMBER:	47
CLAIM_TIME:	13:30
DATED:	030611
DESCRIPTION:	Incident occurred on Highway 42, NY. Damage occurred to front bumper and front right headlight. Airbag was triggered
ESTIMATE_DATE:	03/12/11
ESTIMATE_REPAIR_SHOP_ID:	33675
ESTIMATE_REPAIR_SHOP_NAME:	COMPANY B
ESTIMATE_REPAIR_SHOP_REFERENCE:	VBDA12333
ESTIMATE_TOTAL:	475.69
GENERATE_CLAIM_FORM:	<input type="checkbox"/> True
GENERATE_ESTIMATE_REQUEST:	<input type="checkbox"/> True
GENERATE_INVOICE_REQUEST:	<input type="checkbox"/> True
INJURED:	N
INVOICE_DATE:	03/17/11
INVOICE_REPAIR_SHOP_ID:	33675
INVOICE_REPAIR_SHOP_NAME:	COMPANY B
INVOICE_REPAIR_SHOP_REFERENCE:	VBDA12333
INVOICE_TOTAL:	475.69
POLICY_EFFECTIVE_DATE:	10/10/2011
POLICY_EXPIRY_DATE:	10/10/2010
POLICY HOLDER ADDRESS:	1223 SPRINGFIELD DRIVE, NY
POLICY HOLDER NAME:	JOHN DOE
POLICY HOLDER ZIP:	234567
POLICY_NUMBER:	43398313
SIGNED:	Y
VEHICLE_COLOR:	Silver
VEHICLE_LICENSE:	5566TY
VEHICLE_MANUFACTURER:	Lexa
VEHICLE_MODEL:	Charger
VEHICLE_VIN:	2B3AA4CTX82KP3456

Attachments:

CLAIM_DOCUMENT

ESTIMATE_DOCUMENT

INVOICE_DOCUMENT

Figure 8-14 All data used in the flow

8.4 Business Process Manager step configuration

This section explains the step configuration for BPM.

8.4.1 DBExecute step

For the development of this use-case scenario, we use IBMDB2 Express-C v9.7.4 for Windows. At the time of publication, this edition is available free of charge from the IBM DB2 database server.

By using this release of DB2, we built three database tables. The relationship is detailed in Figure 8-15.

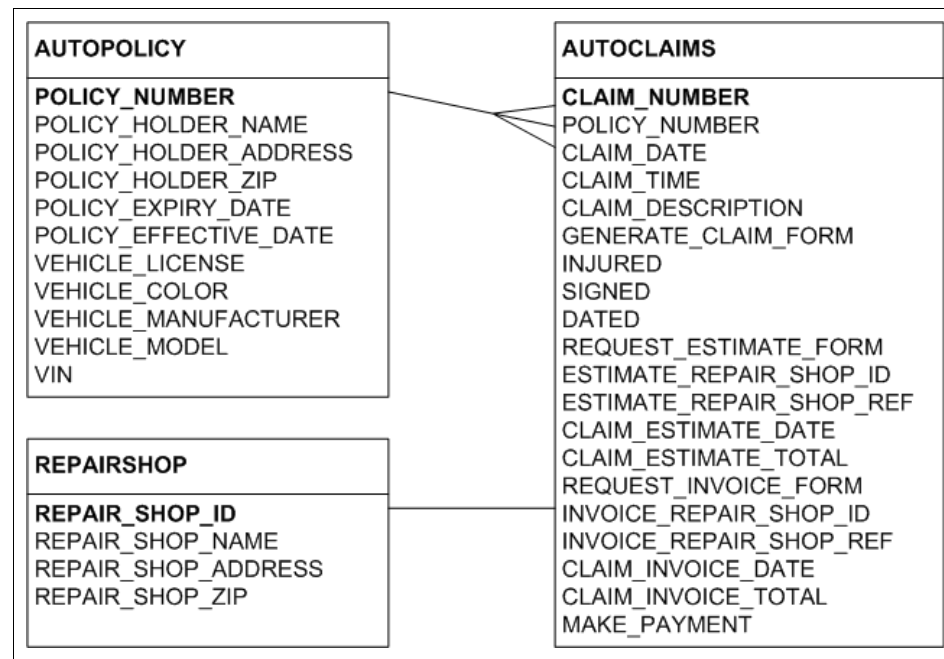


Figure 8-15 Database relationships

The back-end database served as a simulated back-end system, having details about user policies, repair shop information, and so on. In a real-life scenario, the back-end system is more complex and might require integration through the use of web services.

To query the database by using FileNet P8 BPM Tools, we used stored procedures. Usage of a stored procedure removes the need to code SQL into a calling application. Also, because the procedures are stored in the database,

some of the overhead is removed compared to inline SQL queries. For more information, see documentation from your database vendor.

First we configured a database alias so that FileNet P8 can query the database. This task is done in the Process Configuration Console (Figure 8-16). A database alias is created, which is db2 in this example, and the relevant credentials are entered.

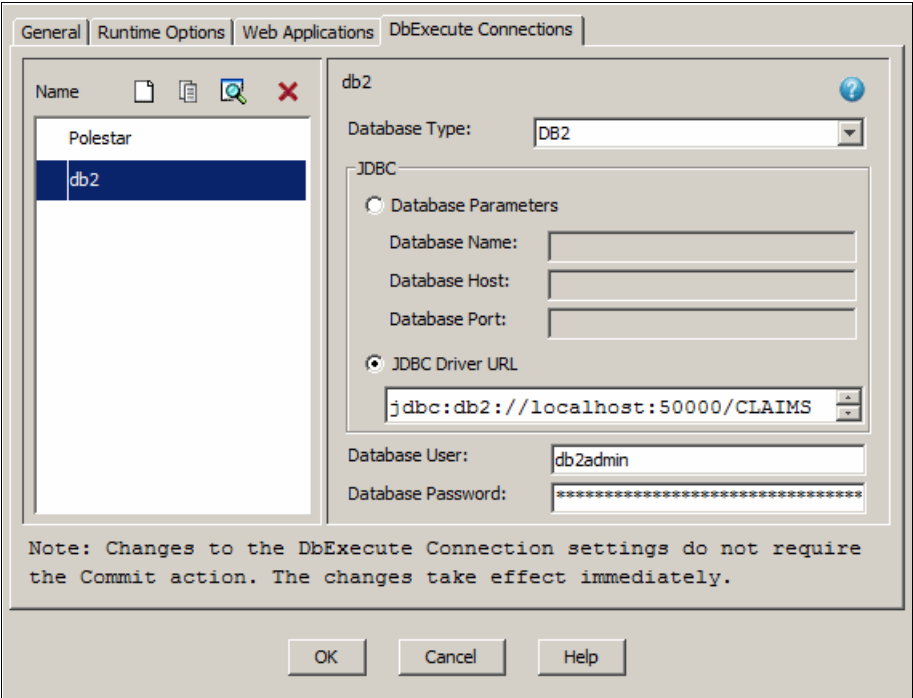


Figure 8-16 Setting a DB2 alias in the Process Configuration Console

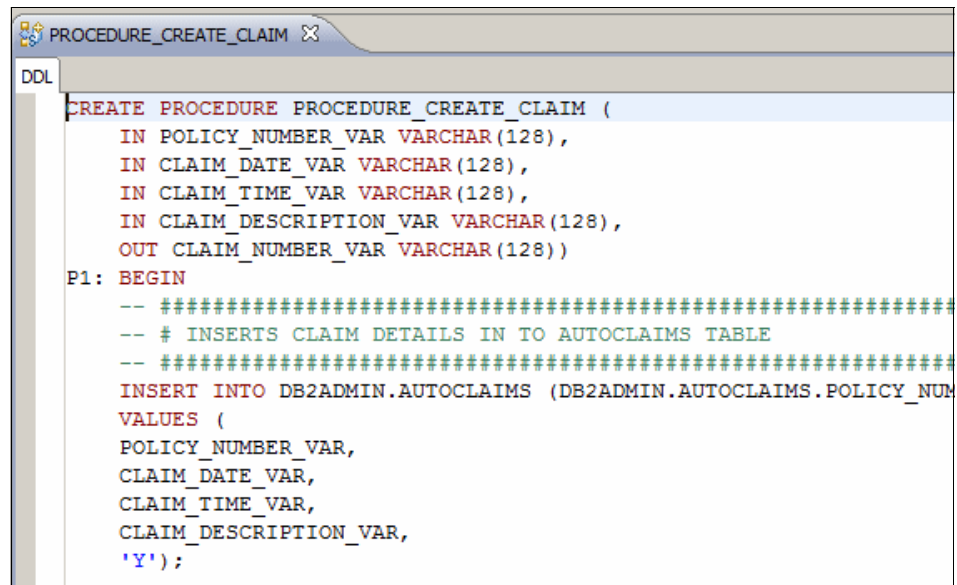
After this alias is created, it can be referenced in the Process Designer to run the stored procedures that were created and stored in DB2. Figure 8-17 shows an example of the setup for an alias.

The screenshot shows the 'Workflow Properties' dialog box for a step named 'Create Claim and Generate Form'. The 'General' tab is selected. The 'Database Connection Alias' is set to 'db2' and the 'Procedure Name' is 'PROCEDURE_CREATE_CLAIM'. Below these fields is a 'Parameters' section with a table listing five parameters: POLICY_NUMBER, CLAIM_DATE, CLAIM_TIME, DESCRIPTION, and CLAIM_NUMBER. The table has columns for an index number and the field name.

#	Field Name
1	POLICY_NUMBER
2	CLAIM_DATE
3	CLAIM_TIME
4	DESCRIPTION
5	CLAIM_NUMBER
6	

Figure 8-17 Calling a stored procedure from the Process Designer

Figure 8-18 shows an example of a stored procedure.



```
CREATE PROCEDURE PROCEDURE_CREATE_CLAIM (
  IN POLICY_NUMBER_VAR VARCHAR(128),
  IN CLAIM_DATE_VAR VARCHAR(128),
  IN CLAIM_TIME_VAR VARCHAR(128),
  IN CLAIM_DESCRIPTION_VAR VARCHAR(128),
  OUT CLAIM_NUMBER_VAR VARCHAR(128))
P1: BEGIN
  -- #####
  -- # INSERTS CLAIM DETAILS IN TO AUTOCLAIMS TABLE
  -- #####
  INSERT INTO DB2ADMIN.AUTOCLAIMS (DB2ADMIN.AUTOCLAIMS.POLICY_NUM
VALUES (
  POLICY_NUMBER_VAR,
  CLAIM_DATE_VAR,
  CLAIM_TIME_VAR,
  CLAIM_DESCRIPTION_VAR,
  'Y');
```

Figure 8-18 Stored procedure used to create a claim in DB2

8.4.2 Conditional step

The conditional step in this process is configured to wait for a specific condition. As shown in Figure 8-19 on page 287, the WaitForCondition monitors the Invoice User Added workflow for its CLAIM_NUMBER to equal CLAIM_NUMBER in the main Auto Claims Process flow. When a match occurs, it copies the values from the triggering flow, which is the data captured by Datacap in this example, to values in the main flow Auto Claims process. This task is carried out in the user interface, which allows for making simple changes without changing the code, for example.

Condition

Workflow Name

Invoice Document Added

Expression Data Type

String

Condition Identifier

CLAIM_NUMBER

Operator

is equal

Expression

CLAIM_NUMBER

Action

Assignments

Name	Expression
INVOICE_DATE	DATE
INVOICE_DOCUMENT	INVOICE_DOCUMENT_IN
INVOICE_REPAIR_SHOP_ID	REPAIR_SHOP_ID
INVOICE_REPAIR_SHOP_NAME	REPAIR_SHOP_NAME
INVOICE_REPAIR_SHOP_REFERENCE	REPAIR_SHOP_REFERENCE
INVOICE_TOTAL	TOTAL

Map

<None>

Close

Help

Figure 8-19 WaitForCondition configuration

8.4.3 Activity step

The activity step has been configured for review by a defined participant. This step can be done for individual users or groups of users. An alternative, which is not used in this process, is to assign the task to a work queue. By assigning the task, any user with access to the queue can run the step, making it much easier to share workloads between teams of users.

Figure 8-20 shows how we expose only those fields that are required at that particular step in the business process. We can also define whether the attribute must be read only or read/write.

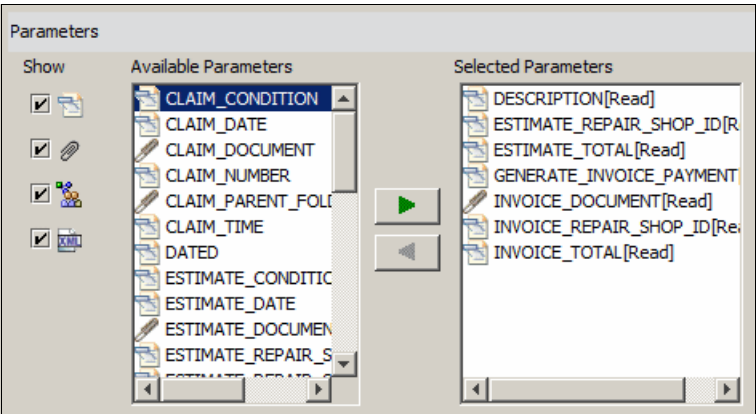


Figure 8-20 Activity step exposing required fields



Best practices and recommendations

This chapter provides best practices and recommendations for building or administering an IBM Production Imaging Edition solution or a pure IBM Datacap Taskmaster Capture (Taskmaster) system.

This chapter includes the following sections:

- ▶ Basic form design and capture
- ▶ Best practices for application development
- ▶ Production Imaging Edition implementation principles

In addition, read Chapter 5, “Designing a production imaging system” on page 139, which offers guidance for designing successful production imaging systems.

9.1 Basic form design and capture

The topic of form design is addressed in various online articles and is beyond the scope of this IBM Redbooks publication. However, this chapter highlights some of the functionality that Taskmaster has available to assist in ensuring good form design. This chapter also highlights general guidelines to follow.

PaperGray font

Datacap ships with its own font called *PaperGray*. This font is in the C:\Datacap\support\fonts directory. Figure 9-1 shows a sample that uses speckled dots to form constrained text boxes.

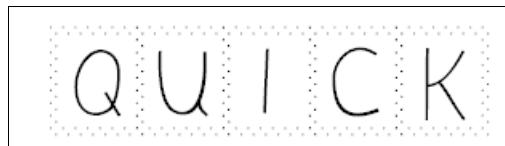


Figure 9-1 *PaperGray* font

Forms that are designed with this font allow segregation of handwritten characters without the use of lines, which can interfere with captured text. This font allows, through use of the image enhance despeckle function, the ability to remove the boxes easily and effectively with minimal impact to the written characters. Although the use of lines enforces character separation, the line removal process, in some situations, can affect the written characters that we are attempting to capture.

Figure 9-2 shows an example of a form that uses the PaperGray font for constrained text boxes. Notice how the boxes have been removed with minimal impact to the text.

Prior to Despeckle

Tel No 0 2 1 7 - 9 6 4 - 0 0
1

National Insurance ☒ Yes ☐ No

After Despeckle

Tel No 0 2 1 7 - 9 6 4 - 0 0
1

National Insurance ☒ Yes ☐ No

Figure 9-2 PaperGray constrained text boxes

Barcodes

Use dimensional (1D) and two dimensional (2D) barcodes where possible. These barcodes aid in identifying a document. They can also carry a large amount of data and, in some situations, all the data you need from the form.

Ideally barcodes must be printed or attached so that they are square with the page. Barcodes that are attached at extreme angles can be difficult to capture.

Colors

Use of color helps to create appealing looking forms. However, the colors must be of a specific range that scanners can drop out (that is, removal of the constrained text box lines that we need).

Most color scanners have a drop-out color that you can specify, which is red or green. These scanners can perform almost the same output as the PaperGray font after despeckling.

Some scanners can produce two images simultaneously. One image is a color image that you can use for export. The other image is a bi-tonal (black and white) TIFF image, with the color removed that we can use for processing.

Always test the color that you want to use before you print large quantities of forms to ensure successful drop out.

Use of colored paper for forms can also affect scanning quality.

Fonts

Ideally use a 10–14 sized font to capture data. Smaller or larger fonts can start to cause issues with the Optical Character Recognition (OCR) engines.

Resolution

The resolution a form that is scanned in can determine the quality of the OCR, Intelligent Character Recognition (ICR), or Optical Mark Recognition (OMR) results. A low-resolution image can make some characters illegible to the OCR, ICR, and OMR engines and cause low confidence or incorrect reads. A higher resolution, although better quality, can pick up additional marks on the form, increasing the number of incorrect reads. This resolution also increases the size of the image that is being stored.

Determine the resolution on a case-by-case basis. However, a general rule is to use 200–300 dpi. Always use at least 300 dpi for OCR/A.

Layout

Use of constrained text boxes for handwritten recognition is important in establishing good results. The text boxes help to define the area where text will reside, the number of characters expected, and potentially the type of character, that is numeric or alphabetic. It also defines the size of characters that is required. These text boxes must be of adequate size so that the person who is completing the form can write legibly. Use the PaperGray font to create constrained boxes (see “PaperGray font” on page 290).

Try to get the person who is completing the form to use black ink and to write in clear, well-formed uppercase characters where possible. Therefore, include instructions in a noticeable area of the form that advise the person completing the form to follow these guidelines. These guidelines can assist in improving the accuracy of the ICR engine.

Ensure that OMR check boxes are of adequate size. Check boxes must not be too small and so close that the person who completing the form selects multiple check boxes. The check boxes must not be too large so that the person only selects a small portion of the box.

Where possible, do not to place constrained text boxes or OMR fields close to the edge of the form. When a form is scanned, a slight misalignment can lead to parts of the image not scanning correctly, resulting in a loss of data.

Constrained text boxes that contain hint characters can also cause issues when scanning if the hint character is not properly removed. Figure 9-3 shows an example along with an alternative method using the PaperGray font.

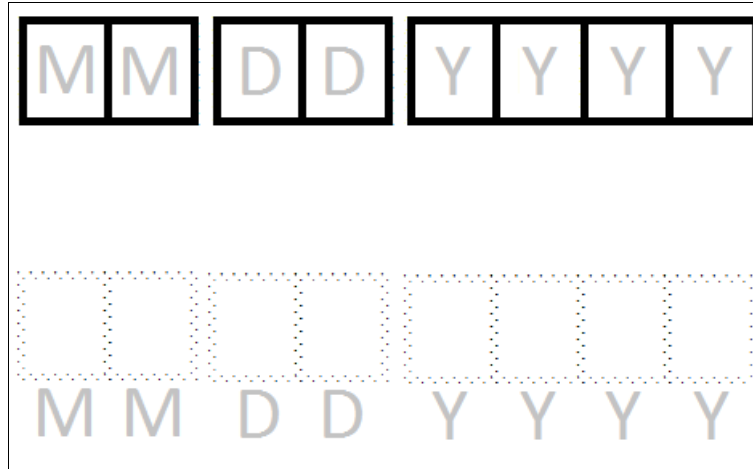


Figure 9-3 Solid and PaperGray constrained boxes

Scanning

To obtain good capture results, use a suitable scanner. When scanning from multiple scanners, the results can differ. The quality from one scanner can be worse than the quality of other scanners. Poor scanning quality can lead to poorly recognized documents. Therefore, make sure that scanners undergo routine maintenance to ensure that they are all working optimally and are not outputting poor quality images.

As indicated earlier, try to not use color on the forms. Although this practice is preferred, certain colors that can be dropped out are permissible. Some scanners can drop out colors at scan time, meaning that they never make it to the original image.

Separator sheets

When using barcode separator sheets, print the separator sheets on lightly colored paper. By using lightly colored paper, they can be easily removed from the scanned batches and reused.

Test any colored paper that you use for separator sheets to ensure that the background appears white. A light blue or yellow color works well with most scanners.

If you are using both document and attachment separator sheets, use of a different color for each sheet allows for easier sorting upon separator removal.

Always use the first generation or barcode separator sheets. Repeated photocopying of the original causes eventual degradation of the barcode to the point where it might become unreadable.

9.1.1 Scanned document verification

You must understand the nature of verification. First, use of Taskmaster does not remove the need for manual verification. You might have to verify a high percentage of the documents that you process. The improvement that you see, however, is increased throughput of documents that might otherwise require manual entry.

For example, typing data from every form requires a user to manually enter all information. Instead, use OCR or ICR to capture this data automatically and identify only the fields that it believes it has had issues recognizing correctly. The advantage is that the verifier, although it might have been seeing a high percentage of the pages scanned and recognized, only has to verify or correct a few the fields on the form. This process drive efficiency in the following ways:

- ▶ You need fewer people to verify the same amount of data as you need for manual entry, reducing head count.
- ▶ You can verify more data with the same number of people, increasing your throughput without increasing head count.

Typically, you get manually completed forms that are, for the most part, completed well (text in uppercase letters, in black ink, and inside the constrained boxes). Alternatively, you get manually completed forms that are, for the most part, completed badly (text in non-black ink, in mixed case, and merged between constrained boxes). As a result, a high percentage of the forms that are completed well go through with only one or two corrections or verifications necessary. The poorly completed forms have a larger number of fields to correct or verify. Therefore, good form design is key to reducing the number of poorly completed forms and thus the amount of verification time.

The Taskmaster thick client verification panel allows usage of keyboard shortcuts with mouse operation. By learning the specific hot keys and shortcuts in this environment, you can obtain greater efficiency in using the system.

Use capture snippets to reduce the need of the operator to search the form for the data to collect. Set these snippets to a size that is suitable for the operator to read easily.

9.1.2 Measuring scan and capture process improvement

The improvement obtained by using Taskmaster can be calculated as the time to process a set of documents with Taskmaster versus the time to process them manually. If you are more productive when using Taskmaster, with less manual intervention, you have succeeded.

You must understand that you are not removing manual intervention. Some form of verification or manual check might be required. Therefore, you are making users more productive, increasing throughput, or achieving both goals.

9.2 Best practices for application development

This section includes some of the best practices and recommendations for application development that we gathered from practitioners in the field. This information is not complete, but provides pointers and guidance based on the real-life experiences of consultants in the field.

9.2.1 Testing an application

When developing and tweaking a system, use VScan to start your batch for the following reasons:

- ▶ When images are scanned into a workflow, they can behave differently based on how they were scanned. If you are adjusting a system to improve recognition, place some problem images in your VScan input directory, and then run the same images over again. If you scan the same images for each run, the differences in the scanning might be responsible for improving your recognition rather than your adjustments.
- ▶ By using VScan, you can quickly create a batch, which negates the time-consuming process of running a set of documents through a scanner.

In many environments, such as test and production environments, you might need to start logging.

To turn on logging for a specific task, complete these steps:

1. Open the Taskmaster Client.
2. Log in to the application for which you want to turn on logging.
3. Select **Settings** → **Workflow** to display the Taskmaster Administrator.
4. Select the task on which you want to invoke logging, for example, VScan. Click the **Setup** button to the right.
5. In the Setup window, select **File** → **Task Settings**.
6. In the Task Settings window, click the **Log** tab. On this page, you can configure the logging levels and file location.

Logs can usually provide a good indication of any issues that are occurring in the system. However, ensure that you set the logs according to the system that is being used. If logging is set too high, it can affect performance.

Keep the flush buffer turned off, unless the batch stops and the log is terminated prematurely.

9.2.2 Capturing data

Define your output requirements from the start. Do not look at the form, and then start to decide what data to extract half way through building your system.

Look at the structure of the system to which you are exporting. What does it require? If this step is not done correctly at the beginning of the process, you can end up doing a lot of work in collecting data with no place to store it after you collect it. You might also omit pieces of data and then have to add them to the project later, increasing the time to deliver. Although this task can be done, knowing what you need up front is much more efficient.

You can capture OCR data from a document by using one of the following methods:

Zone	Defining an area of the form where the data to be captured is located.
Keyword	Using a keyword and then capturing the adjacent text, or using the presence of this keyword to help define the page type.
Regular expression	Use of a regular expression to find data in a particular format such as date, ZIP code or postal code, and so on.
Operator input	Use of an operator highlighting the text to be entered or entering it manually.

For each piece of data that you are asked to collect, decide which method or methods you can use. For data that is not on the form, you can also use database lookups, web services, or other methods to find the data.

9.2.3 Smart parameters

Avoid coding paths as parameters in your actions and scripts. Always read paths from the Application Service or the `settings.ini` file. This way, your application can be easily ported from one machine to another.

Learn to use smart parameters, and use the `rr_` actions from the Rulerunner library to perform various tasks. You can greatly reduce the number of actions that you have to master by using the `rr_` alternatives. For example, `rr_Compare(0,1)` is a good substitute for the `ReturnFalse` action. `rr_Set(Value,@F)` is a good substitute for `DefaultValue(Value)`. The `rr_` actions can perform the same tasks as dozens of other actions and you do not have to learn as many.

9.2.4 Projects

Always copy the foundation and demo applications (Accounts Payable Capture, Flex, MClaims, and so on) before you change them for your own use. If you change the applications in the build, reinstalling them can cause you to lose your changes.

Start new projects with a project that you are familiar with. Altering an application is easier than starting one from scratch.

9.2.5 Actions

Never change an RRX file in the RRX directory. The RRX file can be overwritten with reinstallation. If you want to change one of RRX files, copy the file to your rules folder in your application. If the same name of an `rrx` library is in your rules folder and the RRX folder, the one from the rules folder is loaded.

When writing actions, do not use the same name as another action in another library. Because libraries are loaded at different times during a Rulerunner session, you cannot be sure which action is being called if two libraries are loaded with actions with the same names. The .Net libraries use a namespace at run time. Therefore, the same name can safely exist in the .Net action libraries.

9.2.6 Scripting

You must learn how to write scripts in Taskmaster. The Taskmaster product is flexible and complete, but the included actions do not include the capability to do everything. Therefore by using scripting, you can add additional functionality to the product that was not originally included.

A typical script might take an existing action and add your own code to it to invoke additional logging, routines, or external APIs. The Taskmaster Capture scripting engine can interact with Component Object Model (COM) and .NET objects to take advantage of additional functionality and APIs that are not available with the product. A good example is writing a custom export script and .NET objects to export to an external application.

Because the scripting engine uses VBScript, you can code some portions of the code and then run it from your desktop, which is a good way to test snippets of code quickly outside of the Taskmaster environment.

9.2.7 OMR field configuration

Configuration for use of OMR fields requires use of log files as shown in the following example.

If set appropriately, the `RecogOMRThreshold` action writes the recognized values to the log file. By default, the log files are written to the batch directory. To find the start of the execution within the log file, search for the following line:

```
action RecogOMRThreshold
```

Under this line, you see output similar to the following line:

```
Box1          935,1616,991,1661          2520      411
16.3095238095238
```

This output indicates that the upper left corner of Box 1 is 936 pixels from the left border and 1616 pixels from the top. Also, the lower right corner is 991 pixels from the left border and 1661 pixels from the top. The box covers 2520 pixels. Then 411 pixels, or 16.3095238095238% of the pixels, in the box are black. Keep in mind that the scan was set to scan black and white. Therefore, 2079 (2520 - 411) pixels are white.

If you are unsure about how to set the parameters of the `RecogOMRThreshold` action, run the action with a sample image and set the parameters to any value. Set the log level appropriately, and then read the log file to obtain information similar to the previous information.

If you run this test across multiple OMR fields, multiple pages, or both, you can determine the average OMR threshold for each OMR field.

By using these values, you can determine the thresholds that are needed for the `RecogOMRThreshold` action.

Each task creates an XML file and writes the results of its actions in that file. Furthermore, the `CreateDoc` action creates a data file for each page. Its file name usually starts with the letters “tm,” followed by a series of numbers, and the `.xml` extension. In this file, you find information about all fields on the page. For an OMR field, a status of 0 indicates that the check box is selected, and a status of 1 indicates that the check box is not selected. See Figure 9-4 for a sample of this XML file.

	TYPE : InsuredSignature
	Position : 811,1734,1563,1907
	STATUS : 0
	DensityString : 4
F	SignedDate
	Text value : 062511
	Char confi : 999999
	TYPE : SignedDate
	Position : 929,1984,1262,2048
	STATUS : 0
	IMAGEPATH : \\morpheus\c\datacap\auto_claim\batches\201
	\tm000001.tif
	RecogStatus : 0

Figure 9-4 OMR field used to detect if a signature box has been completed

The `DensityString` field shows the density percentage of the box. If four OMR boxes are available, the density string contains four characters (the same number of characters as the boxes you define). You can compute the density by subtracting 48 from the ASCII code of each character.

For example, the density string for the four boxes shows 0B00. Then the first, third, and fourth boxes have a density of 0%, the character 0 is ASCII 48, and 48 minus 48 equals 0 ($48 - 48 = 0$). The density of the second box, as indicated by a B in the density string, is 18%. The character B is ASCII 66, and 66 minus 48 is 18 ($66 - 48 = 18$).

9.3 Production Imaging Edition implementation principles

A full IBM Production Imaging Edition implementation cannot be delivered without a fair amount of thought, planning, and experience. The Production Imaging Edition bundle is new, but its components have been in existence for a fair amount of time, and methodologies for their deployment have been developed. Also, best practices are described in the product documentation and can be delivered by IBM Lab Services.

Given what was said about use cases, and the efficiency gains to expect, you can use the following guiding principles to help you approach a Production Imaging Edition implementation:

- ▶ Examine the end-to-end business process that is associated to the Production Imaging Edition documents. Try to see what data you need at each step and for whom or what system or process. The idea is to gain an idea about what data is going to be needed, how to get it, and how it is going to be used. You must know what to look for in your documents and external systems.

For example, if we want to process insurance claims, we need to ensure that we can extract insurance policy numbers, customer names, and so on, and check them in the claim administration system. We also need additional documents and pieces of information down the line, such as police reports, damage quotations, and so on.

We must understand how we intend to reconcile the supporting documents with the claim document that drives the process. For example, do we need to send back correspondence with a unique barcode tied to the claim number, so that the claimant can use it when sending the requested documents back?

- ▶ Capture as much data as possible from the business artifacts, that is, from the documents that you have. You want the data to help you make sound decisions, and you need to extract it from the images to make it usable. You do not want to have to search for the data in images and re-enter it manually to do routing, calculations, and so on.
- ▶ Transfer as much relevant information as possible to the Production Imaging Edition workflow. Similarly you want the postcommittal workflow to drive the business process. You also want it to use and to serve as much relevant information as possible to the users or other systems. Therefore, you must pay a lot of attention to designing the data model that is used in the workflow and knowing what data from Taskmaster batches and documents can be fed into it.

- ▶ Normalize the data as early as possible in the process. For example, standardize date formats to help with date calculations, and verify the correct spelling of customer names, vendor names, and so on. Taskmaster and its lookup and verification capabilities are designed for that purpose.
- ▶ As early as possible in the process, integrate with your business systems. The customers, vendors, or product nomenclatures that you use in the business process are likely referenced in an SAP system or other type of business databases. The sooner you can check and validate the data that you have and make them conform to the business systems, the better. It helps everyone down the line and reduces errors. You can use Taskmaster and Business Process Manager, possibly with component integrator, to help in this task.
- ▶ Get documents into the FileNet P8 repository as fast as possible, so that you can realize the benefits associated with the processing of the business documents sooner. Therefore, you must automate the capture process as much as possible. You must look at the types of documents and the forms you are using and then try to standardize them as much as possible, possibly by the use of barcodes.

You must also see what the fastest entry point in the system is, whether it is at a local level or at the central level and who is best placed to supply the data that is needed. This task might mean distributing the capture process to where the documents originate and might be consumed.

When you understand what you want to achieve and have described it on paper, you can start implementing a sandbox system. The idea is to get a test system running quickly with all the major components. Then start setting up the system for one selected business process and its documents and try to implement it end to end.

Typically you want to start setting up the Content repository first because it is the heart of the system. You start with a set of document classes that will be used in a workflow or a folder structure, by the users in security, storage areas, and so on. All of this work is done on the strength of the analysis of the business needs that has been done beforehand.

Then after you define the document classes and the metadata that you need to process the documents, you can configure Taskmaster to capture the documents and metadata that you expect. More likely the analysis of the type of data that can be produced from the documents in Taskmaster and the definition of the document classes in Content Manager will be an iterative process.

By configuring Taskmaster, the following tasks will occur as explained:

- ▶ Creating a Taskmaster application that is dedicated to the business process in question and creating the document hierarchy, the data fields, and zones
- ▶ Configuring Taskmaster tasks and workflow, rule sets and actions, lookups, database feeds, and so on
- ▶ Designing Taskmaster windows and dialog boxes for scan, verify, and data entry tasks
- ▶ Creating users and groups and configuring the functional security that applies to them
- ▶ Configuring the release stage in Taskmaster to commit the documents and the metadata that has been captured to Content Manager and to the workflow
- ▶ Configuring the reporting, activity monitoring, and notification

After a fair amount of testing, iterations, and adjustments on both the Taskmaster and Content Manager, you can start configuring the Production Imaging Edition workflow.

Typically, designing a workflow requires you to look at current business practices. You want to build it as being a better, more efficient way of doing things. Therefore, you must consult with business people over multiple iterations to get the process right.

To help in that process, you use the Business Process Manager Tools, such as the Process Designer and Visio, which can quickly and effectively produce the workflow maps needed. By using the Process Designer, you can create and start testing the workflows. You are likely to need a fair amount of testing and simulations to reach a point where you feel confident that you can start loading the system and do some real-life testing and analysis with Process Analyzer.

Obviously you need to consider many more aspects in an implementation, including integrations with business systems that will likely require the Component Integrator.

Before you can place the system into production, you must perform a fair amount of testing and make the necessary adjustments under various load conditions to achieve the expected results. Bottlenecks might appear between the various components of the system and require remedy. To address bottlenecks, both Taskmaster and Content Manager offer reporting tools that can help to identify them.



Part 3

Advanced technologies

The part provides insight into the advanced technologies that are used to create dynamic applications such as IBM Taskmaster Accounts Payable Capture. It includes an in-depth technical walkthrough to provide invaluable insight to designers in developing and customizing their applications.

This part includes the following chapters:

- ▶ Chapter 10, “Dynamic technologies” on page 305
- ▶ Chapter 11, “Technical walkthrough Accounts Payable Capture” on page 339



Dynamic technologies

This chapter provides insight into software that employs advanced technologies that are used to create dynamic applications such as IBM Taskmaster Accounts Payable Capture and Taskmaster Flex.

This chapter includes the following sections:

- ▶ Introduction to dynamic technologies
- ▶ PageID actions and techniques in dynamic applications
- ▶ FlexID
- ▶ DNA technology
- ▶ Sticky fingerprints
- ▶ Managed recognition
- ▶ CCO Merging
- ▶ FPXML
- ▶ Line item detection
- ▶ Enhanced error messaging
- ▶ Data localization actions
- ▶ Intellocate
- ▶ Flex technology
- ▶ Conclusion

For a technical walkthrough of the IBM Taskmaster Accounts Payable Capture product and to see how these technologies are applied in a real application, see Chapter 11, “Technical walkthrough Accounts Payable Capture” on page 339.

10.1 Introduction to dynamic technologies

Dynamic technologies are useful in solving some of the most difficult challenges in data capture. Most of these technologies were designed to capture data from unstructured and semistructured documents. However, many of the technologies can be added to practically any data capture application to make it more powerful or more manageable.

Dynamic technologies include the following methods and techniques:

PageID actions	Actions to make page identification easier and more manageable.
FlexID	Methods to manually identify pages in the batch.
DNA technology	The Dynamic Natural Analysis (DNA) theory and techniques behind fingerprinting.
Sticky fingerprints	The method for immediately providing location information to multiple examples of the same new fingerprint in a batch.
Managed recognition	A self-monitoring method of calling recognition engines to automatically detect and correct failures in recognition.
CCO merging	The technique of combining all of the CCO files of a document into a single CCO to allow document-level searching of data.
FPXML	A mechanism for storing zones independent of the Setup Document Hierarchy (DCO).
Line item detection	Detecting and displaying repeating data structures on a form.
Enhanced error messaging	An alternative to the standard error messaging employed by validation actions.
Data localization actions	Actions that can detect the localization settings of a machine and programmatically alter the rules execution and data accordingly.
Intellocate	The Accounts Payable Capture learning technique.
Flex technology	Techniques for creating and using data structures that can locate, display, and validate their own data.

This chapter explains each of these technologies and then takes you rule-by-rule through the current version of the IBM Taskmaster Accounts Payable Capture

product to illustrate the use of these technologies in the context of a fully-featured invoicing solution.

10.2 PageID actions and techniques in dynamic applications

Most structured form-driven applications use fingerprinting for the primary page identification technique. However, dynamic applications typically have a document structure that is inherent in the batch composition. There are several reasons for this, but primarily it is because dynamic applications are more document centric in their data structure while structured forms are more page centric. For a given nonstructured document, at application design time, you are unsure whether a piece of data will be found on the first page of the document or on a trailing page.

Dynamic applications usually have three working page types: a required Main_Page and optional Trailing_Page and Attachment page types. Figure 10-1 on page 308 shows the document structure from IBM Taskmaster Accounts Payable Capture.

The Main_Page page type is usually the first page of the document. This page is where all of the data for the entire document resides, regardless of the page from which the data was extracted. Only Main_Page has fields assigned to it.

Other pages of the document are typed as Trailing_Page or Attachment page types. The difference between these additional pages is that pages designated as the Trailing_Page page type are recognized and searched for data as part of the document. Pages designated as an Attachment page type are included in the document, but they are not recognized nor searched by the locate actions.

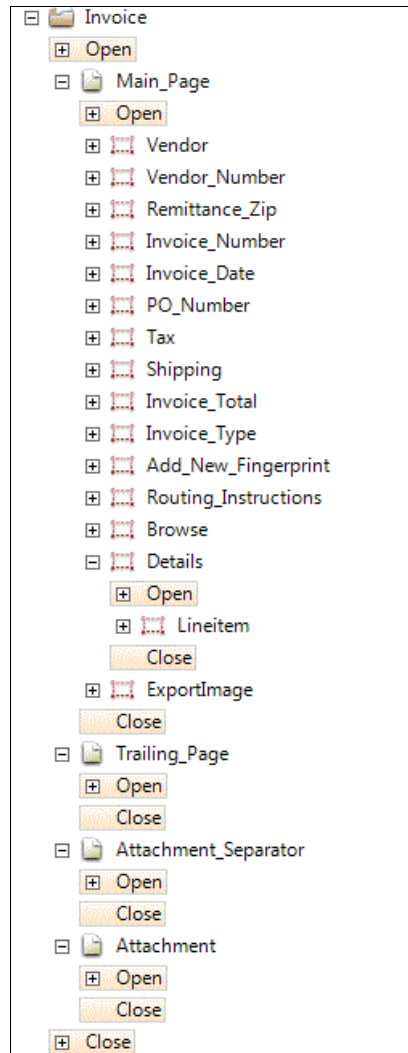


Figure 10-1 Document structure with Main_Page, Trailing_Page, and Attachment page types

Figure 10-2 shows an example where a vendor sends a three-page invoice and attaches a copy of a purchase order, correspondence, or shipping document.

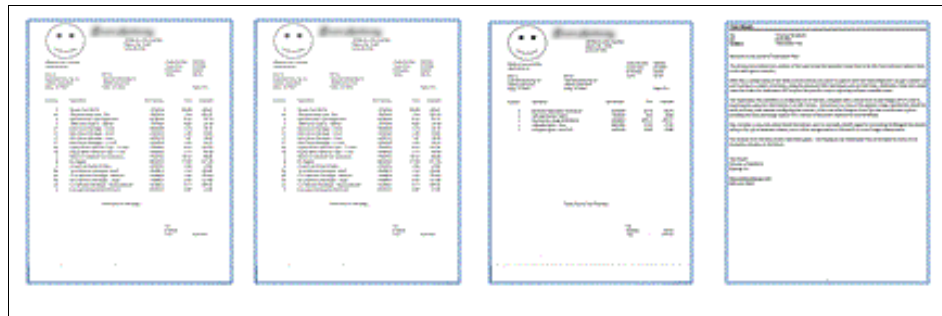


Figure 10-2 Three-page invoice with a correspondence attachment

The first of the three pages is designated as the `Main_Page` page type. The second and third pages are designated as the `Trailing_Page` page type. The page of the attached correspondence is designated as an `Attachment` page type as shown in Figure 10-3.

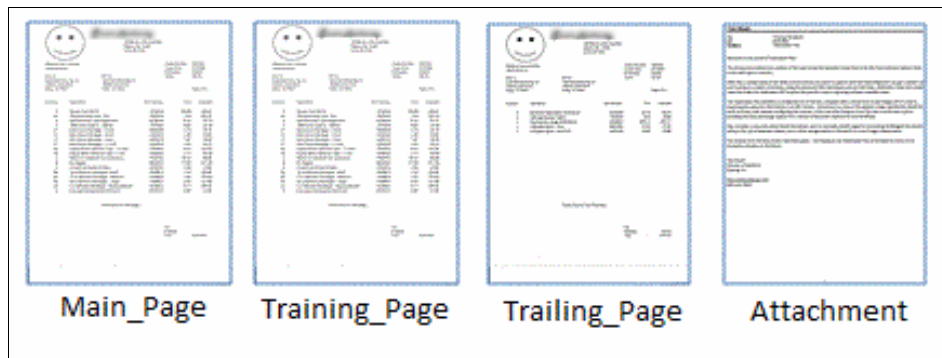


Figure 10-3 Page types of a sample invoice

To aid in page identification, IBM Taskmaster Accounts Payable Capture contains a library of page identification actions called `PageID.rrx`. Inside this library are two actions that are used by dynamic applications to separate the batch into documents.

10.2.1 Page identification by barcode separator

Batches that are scanned into dynamic applications typically need minimal batch preparation.

Most invoices that are processed are a single page. For IBM Taskmaster Accounts Payable Capture and Taskmaster Flex, these single page documents are always placed at the start of a batch. Because every document is a single page, no separation is necessary. When a mail room opens the envelopes that contain the documents, all single page documents go in one stack.

Multiple page documents go into a different stack, with a barcoded separator sheet placed on top of each document. Typically these barcoded sheets are printed on light-colored paper so that, after scanning, the sheets can be easily located and removed for reuse if desired.

You must place an additional separator sheet (Figure 10-4) inside of the document between the actual document pages and any attachment pages that are included.

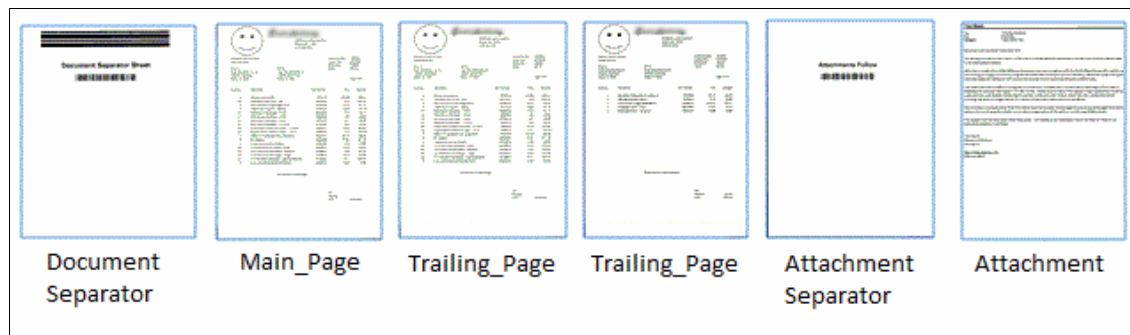


Figure 10-4 Invoice with document and attachment separators in place

In the example with the three-page invoice and the purchase order and shipping document as attachments, an attachment separator page might be inserted after the third page of the invoice, but before the purchase order or shipping document. These pages are placed in a multipage stack, and a document separator sheet is placed on top of them.

When scanning, the only caveat is that, if a batch contains both single-page documents and multipage documents, the single page documents are the first ones scanned. After the actions encounter a document separator, all documents in the batch that follow must also use document separators.

Sample separator sheets are in the C:\datacap\support\seppage directory. This same directory includes the .doc files that you can use to print personalized separator sheets. Be sure that you have a barcode font installed.

The action from the PageID library that is used for such batches is the PageIDbyBCSep action. This action uses two comma-separated parameters, the location of the settings.ini file and the default page type, to name pages in the batch until a document separator is found.

The settings.ini file is typically in the dco_<application name> folder of your project. This action has two pertinent sections as shown in Example 10-1.

Example 10-1 Settings.ini file content related to page identification

```
[PageID_Barcode-PageType]
'Enter the values of barcodes that appear on separator pages, and the
page type to assign to those pages.
ENDDOC=Document_Separator
ATTACH=Attachment_Separator

[PageID_LastType-ThisType]
'Enter the values of every page type, and the page type to be assigned
to unidentified page types that immediately follow them.
Document_Separator=Main_Page
Attachment_Separator=Attachment
Main_Page=Trailing_Page
Trailing_Page=Trailing_Page
Attachment=Attachment
```

The first section, [PageID_Barcode-PageType], lists the values in the barcode on the left side, and the right side is the page type to associate with those barcoded sheets.

The second section, [PageID_LastType-ThisType], documents how you want all nonbarcoded pages in the batch to be named.

When a batch is processed, each page is analyzed to see if it contains a barcode that you specified in the first section. Remember that nonbarcoded pages are given the name that you entered as the second parameter of the action. Therefore, a batch with single-page documents at the start of it is given a page type of Main_Page.

After a separator sheet is encountered, that page is named what is specified in the first section of the .ini file segment in Example 10-1. In this case, the page is named Document_Separator. Any nonbarcoded page immediately after that separator is named according to the listing in the second section of the .ini file

in Example 10-1. For example, because it is the page immediately following a Document_Separator, it is given the Main_Page page type. If another nonbarcoded page follows the Main_Page page type, the second section of the .ini file specifies that it is given the Trailing Page page type. Also, a nonbarcoded page after that Trailing_Page page type is also given a Trailing_Page page type. This pattern continues until another barcoded separator (either Document or Attachment) is found and the “page-after” naming instructions can continue.

10.2.2 Electronic input of documents

Documents that are received into the system electronically do not need separator pages. These documents are typically received as multipage PDF or TIFF files, and you can employ a different technique to name the pages in a batch.

When a file is received electronically, a variable is placed at the page level of the data structure (runtime DCO) that signifies its origin. For multipage files received by a VScan task, this variable is called *ScanSrcPath*.

The PageIDbyVariableChange action takes advantage of this feature by using three parameters to name all of the documents in the batch. The first parameter is the variable to “watch” for change. The second parameter is the type you want to set the first page after the watched variable changes. The third parameter is the type you want to use for the remaining pages in the batch.

For example, consider an email message that contains three TIFF files. At the batch level, you place the PageIDbyVariableChange(ScanSrcPath,Main_Page,Trailing_Page) action.

Each TIFF file is a multipaged invoice. When the images are brought in by actions, the TIFF files are burst into individual TIFF files, and each page has a ScanSrcPath variable that contains a reference back to the source image. If the mail contained three two-page TIFF files, you get a batch with six images, three of them with a ScanSrcPath variable referencing the first multipage TIFF and three images referencing the second multipage TIFF.

The PageIDbyVariableChange action processes the batch. Therefore, when the first source image is referenced by the ScanSrcPath variable, it names the page Main_Page. Then all pages until the ScanSrcPath variable changes are given the Trailing_Page page type.

10.3 FlexID

FlexID is a way to manually identify pages, but some applications use it to perform image quality assurance. FlexID provides a fast, easy way to look at large thumbnails to identify images that might need to be re-scanned and to quickly and conveniently assign page types. This process can be done before or after a page identification is done to the batch. If the process is done before page identification, you normally name only a subset of the pages in the batch, and page identification can use its rules to complete naming of every page in the batch from there.

FlexID is included in a job in the Flex application and in the IBM Taskmaster Accounts Payable Capture product. When using the demo, choose the FlexID option in the job selection you get when you scan.

For example, consider the common way of naming pages within a document in a dynamic application. Normally the first page of the document is named Main_Page, and the other pages in the document are named Trailing_Page. With FlexID, you can identify the first pages of new documents by naming them Main_Page, and the page identification rules that follow name all of the trailing pages for you.

FlexID also provides a way for a user to clone certain images, such as a fax cover page that is in front of three documents that are faxed together. You can copy the header sheet and place it in front of each document with FlexID.

FlexID is available in a thick client or thin client. Figure 10-5 shows the thick-client version.

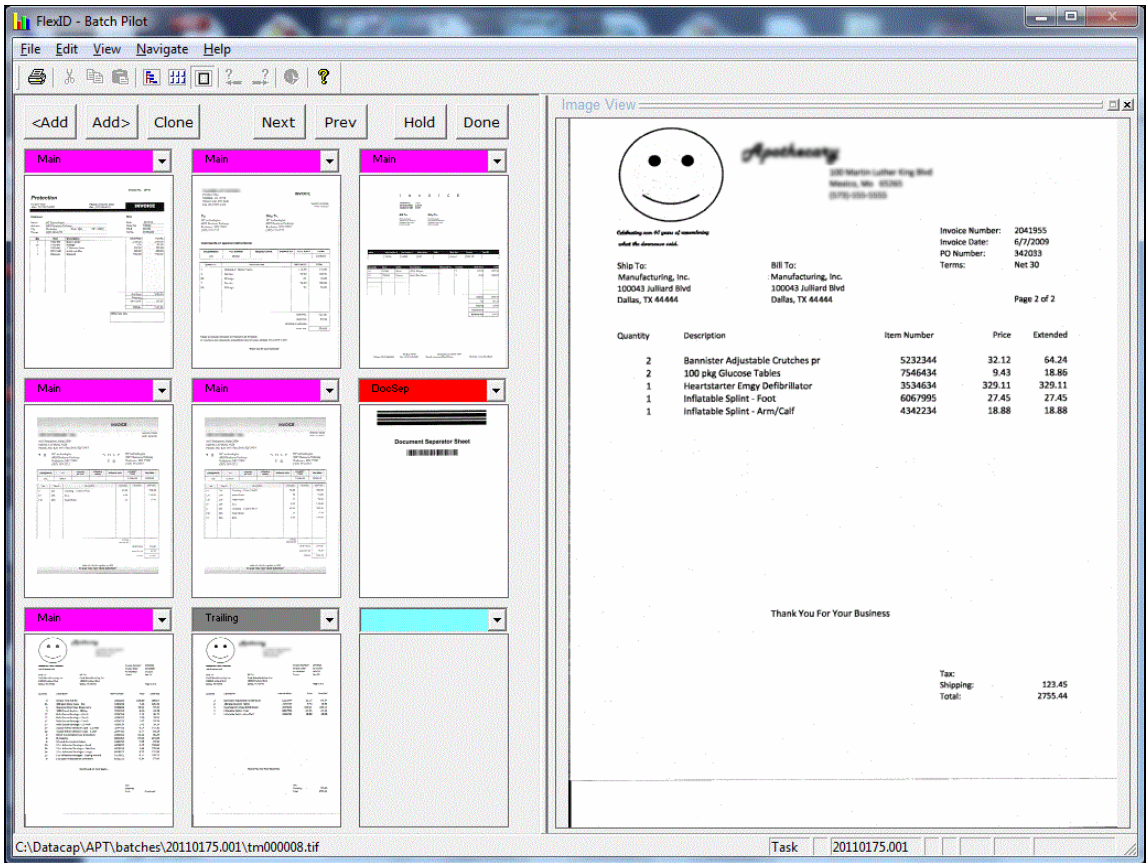


Figure 10-5 Thick client FlexID

10.4 DNA technology

DNA describes the method that is used for image identification based on fingerprint technology. Imagine documents that you are familiar with lying on a table at such a distance that you can identify the document by its layout, without reading any text from the document. For example, you might see your telephone bill or a credit card statement that you are looking at from across the room. Without being close enough to read the text from the document, you can identify it because you are familiar with the layout and because the layout is distinctly associated with that particular document.

By using such a natural analysis, the fingerprint technology processes select images from a batch of images. It also provides three vital pieces of information that are useful in processing each image.

First, the DNA technology notifies you if there was a suitable match between the current image and all of the images currently stored in the fingerprint library. The matching process is done by properly setting up the area of the image to be compared against the library and by defining a confidence threshold that must be met to be considered a match. The “dynamic” part of the technology name indicates the capability of the fingerprint mechanism to add new fingerprints to the library during normal processing if such a match is not found. If a match is not found, the fingerprint automatically goes through a process called *Intellocate*. For information about Intellocate, see 10.12, “Intellocate” on page 335.

Second, if a match is found, the DNA technology returns an identifier, called a *TemplateID*, to the application. This identifier makes it possible to locate previously identified zones on the image where data was found on the image that is in the fingerprint library.

Finally, the DNA technology also returns an offset to apply to those zones to allow for horizontal and vertical movement of the zones to match precisely to the runtime image. When two images go through a mechanical process, such as faxing or scanning, some shifting of the data on the image is common. This offset indicates how far you need to move the reference zones to align properly on the image that they are now associated with.

10.4.1 Fingerprint matching in dynamic applications

When matching fingerprints in dynamic applications, such as IBM Taskmaster Accounts Payable Capture and Flex, the standard fingerprint actions are used similar to way in which they are used in static applications (structured forms). However the usage is different in three key areas.

First, in learning applications, the first parameter to the FindFingerprint action is set to TRUE. With this setting, the fingerprint action creates a fingerprint and adds it to the existing fingerprint library if a suitable match is not found. More stringent matching parameters are associated with this parameter than might normally be applied to the fingerprint facility than if it was used for another purpose, such as, PageID.

In a typical dynamic application, a matching threshold of 80% or higher is common. A balance must be achieved with a threshold that is high enough that documents are not misidentified. However, the threshold must be low enough so that extra fingerprints for the same document are not created unnecessarily. Also, the zoning the search area is normally restricted to the upper part of the

document with a consistent format to allow for variability in the lower portions of the image for line items on invoices, for example.

The second key difference is that the fingerprinting only occurs on key pages of the document. As such, you must override the standard functionality of the fingerprint facility of setting the page type of the matched fingerprint.

With dynamic applications, typically you do a fingerprint match on the first page of a document, to indicate the document type that you are dealing with. Dynamic applications typically name their pages generically, such as *Main_Page*, *Trailing_Page*, and *Attachment*. Typically, only the main page is fingerprinted for identification purposes. The entire data structure for all of the data found on the document is only on the *Main_Page* of the document, although data can be captured from *Trailing_Pages*.

Because the *PageID* of a dynamic application is done before fingerprinting, you must ensure that the page retains its original page type. For example, the rule in Figure 10-6 might run on a page object named *Main_Page*. The page-level rule resets the page type that is returned from the *FindFingerprint* action.

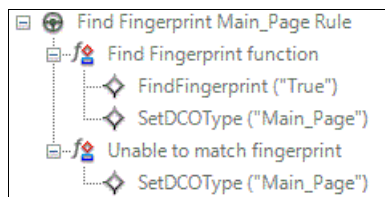


Figure 10-6 A page-level rule for *FindFingerprint* that resets the page type

The *FindFingerprint* action returns *true* or *false* depending on whether it was able to make a match. In either case, you do not want the action to overwrite the page type that you have already assigned in the *PageID* rule set.

After a new fingerprint is processed, you must classify it and save the information, such as zones, that were captured during the processing. For more information about this topic, see 10.12, “Intellocate” on page 335.

10.4.2 Using the *TemplateID*

The *TemplateID* and its associated classification are used to provide valuable information during document processing. The classification is stored in the fingerprint database and is viewable by using the **Zones** tab in Datacap Studio.

In IBM Taskmaster Accounts Payable Capture, the TemplateID is associated with the vendor that supplies the invoice. In Taskmaster Flex, the TemplateID is associated with the Document Class.

The TemplateID also links to systems that indicate you where zonal data is on the form if the fingerprint was previously processed by the system. In cases where a fingerprint match is not found and a new fingerprint is created, no zones are available until it goes through the Intellocate process. For more information about this process, see 10.12, “Intellocate” on page 335.

10.4.3 The offset

The offset returned by the fingerprint mechanism is used to modify the zonal information that is associated with the original fingerprint before it is applied to the runtime image. This offset is derived during the fingerprint matching process to achieve the best possible match of the words and lines on the source fingerprint. Therefore, the offset must be applied to the runtime image before data can be accurately extracted from it.

10.5 Sticky fingerprints

The technology of sticky fingerprints allows for location information. This information is gleaned from processing a new fingerprint at verify time to be applied to other images in the batch that match the same new fingerprint.

For example, the Taskmaster Accounts Payable Capture application contains two different images from the same vendor. If the fingerprint is deleted, the operator encounters the first image and does the data entry on it by using the Click’n’Key capability, supplying the location information for that fingerprint.

When the second image from the same vendor is encountered, a **Sticky Available** button is displayed. When the operator clicks that button, the location information from the previous invoice is applied immediately to the current invoice. The CCO is searched for the header and line-item data, and the document is processed without further operator interaction.

Figure 10-7 shows the first image from the vendor encountered in the batch. Some fields were prepopulated by locate rules.

Verify - Batch Pilot

File Edit View Navigate Help

Lookup Vendor Missing PO

Remittance_Zip

Vendor_Number Invoice_Type

Invoice_Date Invoice_Number Tax

02/26/09

2009/02/26

Invoice_Total PO_Number Shipping

579.42

579.42

Taskmaster

TIO NEW

Details 1(1)

Lineitem 0(0)

Prev Next < Add Add > Del

PO LineNum ItemID Qty

ItemDesc

Price LineTotal

Calculate Blank Find Details View Details

Image View

INVOICE

Invoice # 78002
DATE: 02/26/09

Tracking Systems
2800 Broadway, Suite 1675
Denver, CO 80202-4628
Phone (303) 555-1212 Fax (303) 555-1212

T O Technologies 900 Business Parkway Rochester, MN 55901 (507) 555-1212

S H I P Technologies 900 Business Parkway Rochester, MN 55901 (507) 555-1212

T O

salesperson	PO	SHIPPING METHOD	shipping terms	delivery date	PAYMENT TERMS	due date
191	788914				2/10Net30	03/26/09

qty	item #	description	unit price	discount	line total
6	201	Tracking - Ticket 136432	65.00		390.00
19	206	KCL	6.00		114.00
130	203	Fresh Water	.25		32.50

TOTAL DISCOUNT

SUBTOTAL 556.50

SALES TAX 42.92

TOTAL 599.42

Please all checks payable to: TDS
THANK YOU FOR YOUR BUSINESS!

Task 20110214.014 20110214.014.04 TM000004 NUM

Figure 10-7 The first encounter of this invoice

Figure 10-8 shows the image after the operator clicks the image to locate the missing fields and the upper line item and then clicks the **FindDetails** button.

Verify - Batch Pilot

File Edit View Navigate Help

Lookup Vendor POLR (0)

Remittance_Zip 80202-4628

Vendor_Number TS13687 Invoice_Type PO

Invoice_Date 02/26/09 Invoice_Number E # 70002 Tax 42.92

2009/02/26 70002 42.92

Invoice_Total 579.42 PO_Number 788914 Shipping

579.42 788914

Taskmaster

TIO NEW

Details 1(1)

Lineitem 1(3)

Prev Next < Add Add > Del

PO LineNum ItemID Qty

201 6

201 6

ItemDesc

Trucking - Ticket 138432

Trucking - Ticket 138432

Price LineTotal

65.00 390.00

65.00 390.00

Calculate Blank Find Details View Details

Image View

INVOICE

Trucking Systems INVOICE # 70002 DATE: 02/26/09

2800 Broadway, Suite 1675
Denver, CO 80202-4628
Phone (303) 555-1212 Fax (303) 555-1212

T O Technologies 900 Business Parkway
Rochester, MN 55901
(507) 555-1212

S H I P Technologies
T O 900 Business Parkway
Rochester, MN 55901
(507) 555-1212

salesperson	PO	SHIPPING METHOD	shipping terms	delivery date	PAYMENT TERMS	due date
191	788914				2/10Net30	03/26/09

qty	item #	description	unit price	discount	line total
6	201	Trucking - Ticket 138432	65.00		390.00
19	206	KCL	6.00		114.00
130	203	Fresh Water	.25		32.50
TOTAL DISCOUNT					
SUBTOTAL					556.50
SALES TAX					42.92
TOTAL					579.42

Please all checks payable to: TDS
THANK YOU FOR YOUR BUSINESS!!

Task 20110214.014 20110214.014.04 TM000004 NUM

Figure 10-8 The same invoice after the operator uses Click'n'Key and FindDetails

Figure 10-9 shows the second invoice from the same vendor in the batch. Notice that this invoice is different, including different data and a different number of line items. You must click the **Sticky Available** button in the upper part of the pane, so that the zonal information from the first invoice can be immediately applied to this invoice.

Verify - Batch Pilot

File Edit View Navigate Help

Page 1 of 1 **Sticky Available** Document 5 of 6

Lookup Vendor Missing PO

Remittance_Zip

Vendor_Number Invoice_Type

PO

Invoice_Date Invoice_Number Tax

02/26/10 2010/02/26

Invoice_Total PO_Number Shipping

1551.39 1551.39

Taskmaster

TIO NEW

Details 1(1)

Lineitem 0(0)

Prev Next < Add Add > Del

PO LineNum ItemID Qty

ItemDesc

Price LineTotal

Calculate Blank Find Details View Details

Image View

INVOICE

Trucking Systems

2800 Broadway, Suite 1675
Denver, CO 80202-4628
Phone (303) 555-1212 Fax (303) 555-1212

INVOICE # 10001
DATE: 02/26/09

T O Technologies 900 Business Parkway Rochester, MN 55901 (507) 555-1212

S H I P Technologies 900 Business Parkway Rochester, MN 55901 (507) 555-1212

T O

salesperson	PO	shipping method	shipping terms	delivery date	PAYMENT TERMS	due date
195	678430				2/10Net30	03/26/10

qty	item #	description	unit price	discount	line total
10	201	Trucking - Ticket 138432	70.00		700.00
130	204	Brine Water	.70		91.00
130	203	Fresh Water	.25		32.50
19	206	RCL	6.00		114.00
6	201	Trucking - Ticket 146611	65.00		390.00
130	203	Fresh Water	.25		32.50
19	206	RCL	6.00		114.00
TOTAL DISCOUNT					
SUBTOTAL					1474.00
SALES TAX					77.39
TOTAL					1551.39

Note all checks payable to **PBS**
THANK YOU FOR YOUR BUSINESS!

Task 20110214.014 20110214.014.05 TM000005 NUM

Figure 10-9 The second invoice from the new vendor in the batch

Figure 10-10 shows this invoice after clicking the **Sticky Available** button. The location information from the first invoice is applied to this invoice, and all of the fields and line items are populated with data in one click.

Verify - Batch Pilot

File Edit View Navigate Help

Page 1 of 1 Document 5 of 6

Lookup Vendor Invalid PO

Remittance_Zip 80202-4628

Vendor_Number TS13687 Invoice_Type

Invoice_Date 02/26/10 Invoice_Number # 28100 Tax 77.39

2010/02/26 28100 77.39

Invoice_Total 1551.39 PO_Number 678430 Shipping

1551.39 678430

Taskmaster TIO NEW

Details 1(1)

Lineitem 1(7)

Prev Next < Add Add > Del

PO LineNum ItemID Qty

201 10

ItemDesc

Trucking -- Ticket 138432

Trucking - Ticket 138432

Price 70.00 LineTotal 700.00

70.00 700.00

Calculate Blank Find Details View Details

Invoice_Number Value is too short. Minimum field length is:2 Task 20110214.014 20110214.014.05 TM000005 NUM

Image View

INVOICE

Trucking Systems

2800 Broadway, Suite 1675
Denver, CO 80202-4628
Phone (303) 555-1212 Fax (303) 555-1212

INVOICE # 70002
DATE: 02/26/10

T O Technologies 900 Business Parkway Rochester, MN 55901 (507) 555-1212

S H I P T O Technologies 900 Business Parkway Rochester, MN 55901 (507) 555-1212

salesperson	PO	SHIPPING METHOD	shipping terms	delivery date	PAYMENT TERMS	due date
195	678430				2/10Net30	03/26/10

qty	item #	description	unit price	discount	line total
10	201	Trucking -- Ticket 138432	70.00		700.00
130	204	Brine Water	.70		91.00
130	203	Fresh Water	.25		32.50
19	206	RCL	6.00		114.00
6	201	Trucking -- Ticket 146611	65.00		390.00
130	203	Fresh Water	.25		32.50
19	206	RCL	6.00		114.00
TOTAL DISCOUNT					
SUBTOTAL					1474.00
SALES TAX					77.39
TOTAL					1551.39

Note all checks payable to: **POB**
THANK YOU FOR YOUR BUSINESS!

Figure 10-10 The second invoice after the Sticky Fingerprint technology is applied

10.6 Managed recognition

Recognition engines are expected to attempt to recognize any image that is sent to them. Taskmaster employs many different recognition engines, some of which are more suited for some types of images than others. Over the years, experience has shown that monitoring the recognition engines closely is prudent. Therefore, Taskmaster employs a technology called *managed recognition*.

Figure 10-11 illustrates the managed recognition concept.

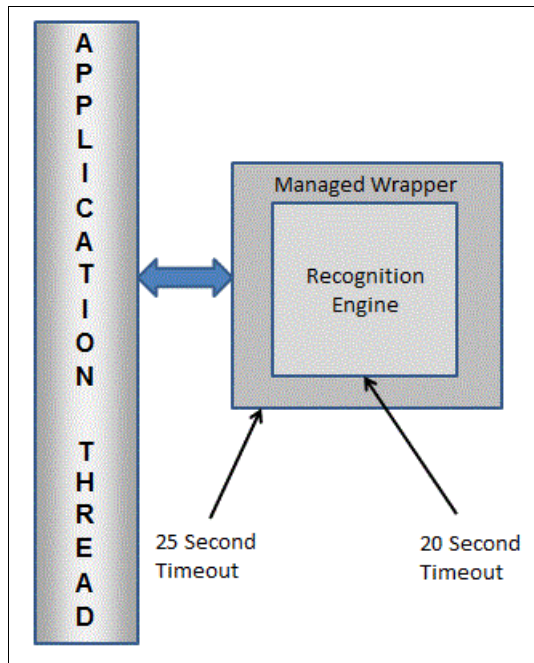


Figure 10-11 Conceptual drawing of managed recognition

With managed recognition, the recognition engine is placed outside of the current thread and monitored by Taskmaster. When recognition occurs, if the recognition engine is successful in recognizing the image, it returns the data directly to the main application thread for further processing.

If the recognition engine is unable to recognize the image, and exceeds its timeout value, it returns to the main application for another try with a longer timeout.

If the recognition engine hangs and is unable to monitor its time, the Managed Wrapper detects this problem. Then it shuts down the engine immediately, reinitializes it, and tries a second time with a longer timeout.

These timeouts are configurable and might need to be changed based on the speed of the machine running recognition, or the complexity of the images being recognized.

If both attempts at recognizing an image fail, the document is automatically marked for deletion from the current batch. Then a notice is set to administrative personnel along with images for further review.

10.7 CCO Merging

As was mentioned previously, dynamic applications place their data structure only on the Main_Page of the document. However, when searching for data, they search both Main_Page and Trailing_Page page types.

To accomplish this task, you must use a technology called *CCO Merging*. When you use CCO Merging, the CCO file for the Main_Page is combined with the CCO files from all of the Trailing_Pages. Also, a multipage CCO (MCCO) is created to replace the single-page CCO that is associated with the Main_Page. If the single-page CCO from the Main_Page must be used later in the process, your application must take steps to back it up and restore it.

The merging of CCO files is done by using an action called *MergeCCOs_ByType*. This action is typically done at the document level after a document structure is created and before the document is searched for data. IBM Taskmaster Accounts Payable Capture and Taskmaster Flex run this action in the Locate rule set.

`MergeCCOs_ByType(Main_Page,Trailing_Page)` merges all of the Trailing_Page CCO files onto the Main_Page CCO file and replaces the Main_Page CCO file with the result. After this task is done, locate rules search the entire CCO file for the data that you require.

When data is found on a Trailing_Page, the coordinates are in relation to the Main_Page of the document. For example, consider two CCOs for pages that are 3300 pixels in length. The resultant MCCO is 6600 pixels in length. Also, something that is 500 pixels down on the second page shows a vertical coordinate of 3800 pixels (3300 pixels from page 1 and an additional 500 pixels down on page 2).

These coordinates cannot be displayed correctly in a verify panel that shows a single TIFF at a time. Therefore, these raw coordinates must be adjusted back to page coordinates before verification.

This step is done by the `MCCOPositionAdjust()` action. This action runs after all of the data has been programmatically extracted by Locate rules.

After `MCCOPositionAdjust` runs, an `IMAGEFILE` variable is displayed in the field that contains a reference to the actual page on which the data was found. All of the positions are adjusted related to that page. In the example used with the two pages, data found on the second page might have an `IMAGEFILE` variable referencing the actual TIFF from which the data was extracted. The 3800 vertical coordinate might be adjusted back to 500 pixels.

10.8 FPXML

Taskmaster has two places where it is possible to store the zonal information for a field. By default, applications store the field position information in the Setup DCO. When you have a field that occurs on multiple documents, the Setup DCO stores the position information in tags that reference back to the Template ID, such as in Example 10-2.

Example 10-2 Snippet of the Setup DCO

```
<V n="Pos1003">2134,1179,2249,1214</V>
<V n="Pos1010">2226,1098,2381,1139</V>
<V n="Pos1016">2177,1571,2315,1615</V>
<V n="Pos1002">2268,1612,2392,1646</V>
<V n="Pos1005">2178,1103,2315,1145</V>
```

In the snippet of the Setup DCO in Example 10-2, the field in question is on five different documents. The TemplateID of each Main_Page is encoded in the tag itself, such as Pos1003. Also, the comma-separated list contains the coordinates of the field in the sequence X1,Y1,X2,Y2.

As you can imagine, if you have an application such as IBM Taskmaster Accounts Payable Capture, tens of thousands of Template/Position lines might exist for every field that is to be captured zonally. This number of lines can result in excess time in rewriting the Setup DCO each time a new fingerprint is automatically added to the system.

An additional consideration is the portability of the Setup DCO from one system to another. References to the rules are also stored in the Setup DCO. Consider a situation where you change the rules on a test or development system and then move the Setup DCO to a production system. In this case, you lose the field positions of any field that was automatically added to the production system because the file was copied to a test or development system.

For this reason, use the FPXML system to store zonal information when developing dynamic applications. FPXML stores the positions for each fingerprint in a separate XML file. The file is stored in the fingerprint directory alongside of the fingerprint TIFF and the fingerprinted CCO.

When it is time to read the zones for a given image, the ReadZonesFPX action from the FPXML action library is used. This action is the FPXML equivalent to the ReadZones action from the Zones action library, but it reads the zones from the FPXML file instead of the Setup DCO.

When writing the zones with features such as Intellocate, you use a different action other than the ones that write to the Setup DCO. The WriteZonesFPX action is a direct replacement for iloc_SetZones and iloc_SetDetailZones, again with the exception that it writes zones to an FPXML file instead of the Setup DCO.

To set up Datacap Studio to use FPXML for your application, FPXML requires an entry in the .app file of the application such as the one shown in Example 10-3.

Example 10-3 Entry in the .app file

```
<k name="dco_FlexID">
  <k name="UseFPXML" v="true"/>
</k>
```

When this entry is successfully completed, new fingerprints added to the system have a file in the fingerprint directory with the naming convention <TemplateID>.xml. This file includes a tag for each field in your DCO, followed by the position at which the field data can be located for that fingerprint.

A facility is built into the Fingerprint Maintenance Tool to convert positions from FPXML to the Setup DCO and vice versa. The Fingerprint Maintenance Tool also uses FPXML to export fingerprints from a system to be imported into another system.

10.9 Line item detection

Line item detection is used to capture areas of repeating data in a document, such as the line items often found on invoices or purchases on a credit card statement.

Figure 10-12 shows sample line items from an invoice. This invoice has three line items. Taskmaster can capture an unlimited number of line items in the entire document, even multipaged documents, by using the CCO merging technology.

Quantity	Description	Item Number	Price	Extended
3	Deluxe First Aid Kit	3332223	129.89	389.67
45	100 pack Daisy Cups - 4oz	6423216	7.23	325.35
5	Stainless Steel Tape Dispensers	8738224	19.55	97.75

Figure 10-12 Sample line items from an invoice

To understand how Taskmaster deals with line items, you must be aware of how repeating structures are stored, zoned, captured, and filtered.

10.9.1 Storing repeating structures

Repeating structures are stored in a detail field structure. A detail field is typically a field that is placed directly as a child to a page in the DCO, but contains a structure of subfields beneath it to hold the repeating data.

Figure 10-13 shows the detail section from IBM Taskmaster Accounts Payable Capture. In this application, the coordinates of each line of the invoice are stored as a Lineitem type field. An unlimited number of these fields can exist when the repeating structure is filled with an invoice that is being processed. Each line item has six subfields of its own that contain the actual data we want to capture off the line.

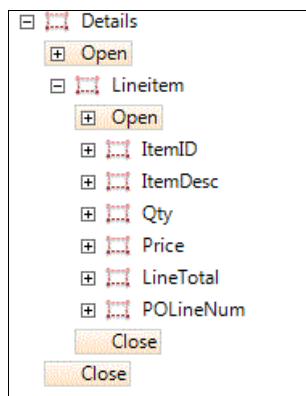


Figure 10-13 Detail structure where all of the line item data is stored

The subfield that is immediately under (is a child of) the Details field contains a zone to the entire line that you want to capture. This line is not always a physical line on the page. Some repeating structures use multiple physical lines on the image to describe a single-line item.

Figure 10-14 shows an invoice with multiple physical lines for each line item you will capture. The ItemID is on the second physical line of each line item, but is stored in the same document structure as invoices with single physical line items.

Specification of commodities			Quantity	U/M	Unit price USD
Line	Description	Ship Date	Rel. No.		
1	BALED CEILING TILE WOOL			22.310 TN	192.0000
	Product No. 2000002	12/05/2010	40		
	Line Item P.O. 54348				
2	BALED CEILING TILE WOOL			22.570 TN	192.0000
	Product No. 2000002	12/05/2010	50		
	Line Item P.O. 54348				

Figure 10-14 Multiple physical lines documenting a single line item

10.9.2 Zoning the detail structure

Each field in the structure needs a zone. With the use of Intellocate, this zoning is initially captured at run time by the verifier clicking the subfield components with the first instance of the document that is processed through the system. From these subfield components, the line item position itself is calculated. Then the detail field is calculated from the top of the line item to the bottom of the CCO.

Under normal conditions with a dynamic application, you only need this automatic zoning of the detail structure. However, for some applications that are form driven, you might choose to zone detail fields manually by using Datacap Studio.

Important: Use care when zoning detail structures on the **DStudio Zones**. Thoroughly test your zones after you complete this task.

To zone the detail structure on the **DStudio Zones** tab, complete these steps:

- 1. Zone each subfield by drawing a box around each item as shown in Figure 10-15.

Quantity	Description	Item Number	Price	Extended
3	Deluxe First Aid Kit	3332223	179.89	539.67
45	100 pack Daisy Cups - 4oz	6423216	7.23	325.35
5	Stainless Steel Tape Dispensers	8738224	19.55	97.75

Figure 10-15 Manually zoning the line item subfields

2. Zone the line item field by drawing a box around it, but be sure that it does not overlap any of the subfields. This step can be difficult because you cannot see the subfield zones while you are doing the zoning. Keep the zone tight around the subfields, but allow for more liberal zoning around the entire line item as shown in Figure 10-16.

Quantity	Description	Item Number	Price	Extended
3	Deluxe First Aid Kit	3332223	129.89	389.67
45	100 pack Daisy Cups - 4oz	6423216	7.23	325.35
5	Stainless Steel Tape Dispensers	8738224	19.55	97.75

Figure 10-16 Manually zoning the Lineitem field

3. Zone the detail area. Again, the line item must be inside of the details zone that you set. Therefore, be liberal in your zoning to allow space around where you zoned the line item as shown in Figure 10-17.

Quantity	Description	Item Number	Price	Extended
3	Deluxe First Aid Kit	3332223	129.89	389.67
45	100 pack Daisy Cups - 4oz	6423216	7.23	325.35
5	Stainless Steel Tape Dispensers	8738224	19.55	97.75
3	1000 Count Aspirin - 100mg	3734353	8.26	24.78
17	Rolls Gauze Bandage - 4 inch	8534554	2.16	36.72
17	Rolls Gauze Bandage - 3 inch	4334522	1.99	33.83
17	Rolls Gauze Bandage - 2 inch	4334523	1.67	28.39
17	Rolls Gauze Bandage - 1.5 inch	4334524	1.42	24.14
24	3-pack White Adhesive Tape - 1.5 inch	2344133	6.14	147.36
12	4-pack White Adhesive Tape - 1 inch	2344132	5.77	69.24
2	OSHA 4.3 compliant ear protectors	2332555	43.12	86.24
8	XL Goggles	8456454	27.66	221.28
4	12 pack Ammonia Inhalers	6342233	5.99	23.96
36	12 ct Adhesive Bandages - Small	4428813	4.19	150.84
36	12 ct Adhesive Bandages - Medium	4428814	5.44	195.84
24	12 ct Adhesive Bandages - Large	4428815	6.32	151.68
12	5 ct Adhesive Bandages - Gaping Wound	9110911	9.11	109.32
9	5 oz tube Antibacterial Ointment	6332212	6.34	57.06

Continued on next page...

Figure 10-17 Manually zoning the Detail field

Notice that you only need to zone the top line item and the subfields, but the detail field must extend over all line items.

If you are working with a project that has a fixed number of line items, such as on a form, the bottom of the details section can be set exactly. However, if you are zoning something that has a variable number of line items, the bottom of your detail zone must be reset programmatically to the bottom of the CCO. This resetting is done at run time by using the ZoneBottom_ImageBottom action. For more information, see 10.9.3, “Capturing the detail structure” on page 329.

10.9.3 Capturing the detail structure

Capturing the detail structure is done in three steps. This task is typically associated with the Locate rule set, where the header data in a dynamic application is found.

At the detail field level, you must first dynamically set the boundaries of the detail zone. If you are working with a form that has a fixed number of detail lines, the zone that you set in Datacap Studio does not need to be programmatically altered. However, if you are working with a document that has an unknown number of line items that can span many pages, adjust your detail zone so that all of the potential line items are contained in it.

This task is done by using the following rules from the Zone action library:

- ▶ ZoneTop_ImageTop()
- ▶ ZoneBottom_ImageBottom()
- ▶ ZoneRight_ImageRight()
- ▶ ZoneLeft_ImageLeft()

These four actions extend your detail zone to the entire document. Do not worry if you get additional text that is not a line item in the zone. It is filtered out later.

The action that creates all of the line item child fields is the ScanDetails() action. This action creates a Lineitem for every physical line in the detail zone. If 100 physical lines are in the zone, it creates 100 Lineitem children fields, with a unique ID for each one (LineitemXX), where XX is a sequential number starting with Lineitem1. The actual name represents what you named your child field in the Setup DCO. Each line that is created has its upper-left corner set at the X coordinate that is assigned when zoning the line item. The Y coordinate is the top of the actual line detected by the coordinates of the line in the CCO. The lower-right corner of the line zone is set from this point with the width and height of that the Lineitem zone.

If your Lineitem field is defined to capture more than one physical line, the result is multiple overlapping zones, which are not a problem. Only the zones that have complete line items are kept after the filtering.

After this process is done, you usually have many line-item fields. Although each of these fields has a different ID, they all have the same type. Therefore, a rule defined in Datacap Studio to run on that object type is run for each line item.

The ScanLineItem() action used on each line item. This action creates the subfields for each line item, in relation to the position of each line item. The relative position of each field in the line item is determined by the position in

which the field is defined in the initial line-item zone that is defined by Intellocate or by manual zoning in Datacap Studio.

When this task is done, the subfields of each line item must be positioned where you expect line item data to be on each line item. To pull the data from the CCO into the fields, you usually use the `PopulateZNLinItemField()` action from the Zones action library. In rare circumstances, you can use other locate actions, such as the `FindRegExInZone` action from the Locate action library. However, normally capturing all of the data in the zone is appropriate.

Again, you can expect many more lines in the processing than you have actual line items in the detail area. All of the header and footer information from all of the pages in the CCO are now zoned and captured as line items. However, only a subset of those zones is retained as line items through a process called *filtering*.

10.9.4 Filtering line items

Filtering line items is a two-stage process that requires two rule sets. The two rule sets act in concert with each other and are typically called *Clean* and *Filter*.

The goal of the filtering process is to remove all of the line items that you do not want from the captured set of line items. To accomplish this task, you must identify certain characteristics about some of the fields of the data set.

At the time of cleaning and filtering the lines, remember that all physical lines are represented in the data structure. Also keep in mind that, in the example in this book, we have zoned each line and pulled any values that were in the zones in which we are looking for particular data.

Figure 10-18 shows the zoned data. Recall that we are zoning every physical line item in the document as though it was a line item. The strategy is to look at key fields where we know that we want a specific data type.

Dallas TX 3443	Dallas, TX /3443			Page 1 of 2
Quantity	Description	Item Number	Price	Extended
3	Deluxe First Aid Kit	3332223	129.89	389.67
45	100 pack Daisy Cups - 4oz	6423216	7.23	325.35

Figure 10-18 Zoned lines

Table 10-1 shows the data that is captured in each of these fields. (Shown in a grid for convenience, the DCO is stored in an XML file.)

Table 10-1 Data pulled from the zoned lines

Qty	Description	ItemID	Price	LineTotal
TX	Dallas,TX 73443			2 of 2
	Description	Number	Price	Extended
3	Deluxe First Aid Kit	3332223	128.89	389.67
45	100 Pack Daisy Cups	6423216	7.23	325.35

The cleaning process analyzes the data in each field and then blanks it out if it is not the correct data type. In Table 10-2, the Qty field was analyzed and left blank if it did not contain a number. Also the Price and LineTotal fields were left blank if they did not contain a currency value.

Table 10-2 Data left blank during the cleaning process

Qty	Description	ItemID	Price	LineTotal
	Dallas,TX 73443			
	Description	Number		
3	Deluxe First Aid Kit	3332223	128.89	389.67
45	100 Pack Daisy Cups	6423216	7.23	325.35

Next, a filter is applied to remove all line items that do not have values in the specified number of fields. In this example, you create filter such as the one shown in Example 10-4.

Example 10-4 Filter to remove all line items that do not have values

```
CheckSubFields(('Qty' AND 'LineTotal') OR ('Price' AND 'LineTotal') OR  
( 'Qty' AND 'Price' )
```

In this example, we are looking for valid data in only two of the three fields that we analyzed for data types. This way we have some flexibility if we run across an invoice that does not list the quantities or the price, or if the recognition on the field came back incorrectly for some reason.

At the end of this process, you are left with only the last two lines in the example, the nonlinear items have been filtered out (Table 10-3).

Table 10-3 Lines filtered

Qty	Description	ItemID	Price	LineTotal
3	Deluxe First Aid Kit	3332223	128.89	389.67
45	100 Pack Daisy Cups	6423216	7.23	325.35

10.10 Enhanced error messaging

When a validation action fails, it sets an error message at the field level that is displayed in the status bar of a verification panel. When the action fails, it sets the error message according to the specific action that fails. The result is that, in certain situations, the operator does not have a complete understanding of everything that is wrong with a particular field. Consider the example in Figure 10-19.

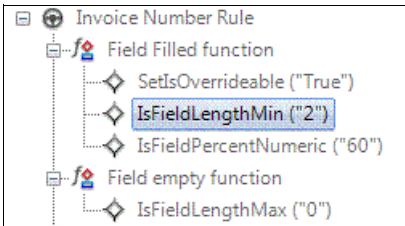


Figure 10-19 A validation rule that sets the field-level error messages

This type of rule is common as a validation. The first function checks to see if the data in the field is at least 2 characters long and at least 60% numeric. If this function fails, the second function in the rule indicates that it can also pass validation if the field is blank.

Consider what happens if someone types a value of A1 in the field. It passes the action concerning the minimum length, but fails the action that indicates that it must be at least 60% numeric. The field-level error message is changed to “The data must be at least 60% numeric.”

Then, the rule runs the second function that indicates it can also pass validation if it has a maximum length of 0. This function also fails. Therefore, the value of the field-level error message is changed to “The field must have a maximum length of 0 characters.”

Obviously, the error message can be misleading to a data entry operator.

The solution to this problem is to use the enhanced error message technology in IBM Taskmaster Accounts Payable Capture. With this enhancement, you can control the error message that is ultimately displayed to the user in a dialog box when validation is attempted during verification. For an example, see the AddToErrorMsg action in Figure 10-20.

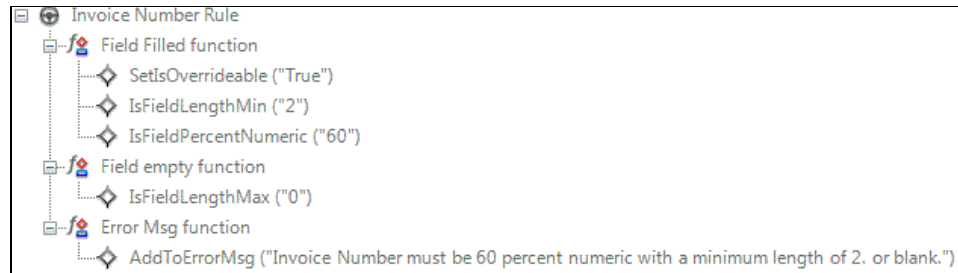


Figure 10-20 AddtoErrorMsg action

This error message is saved at the page level in a variable that is appended to all other error messages from the document. This way the operator has a complete understanding of all errors that are found during validation for the entire document as shown in Figure 10-21.

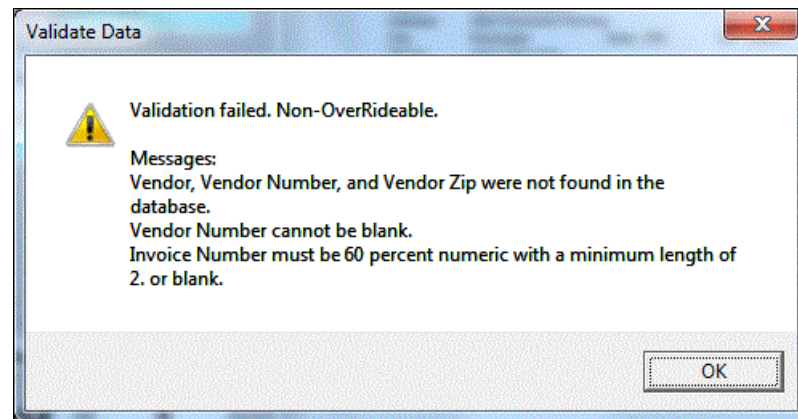


Figure 10-21 A page-level enhanced error message

10.11 Data localization actions

Taskmaster has built-in language support for various languages. However, some actions are needed to handle localized dates and numeric formats. This list grows because opportunities are available in various locales and new formats must be handled. The strategy is to detect the localization options of the computer running the application. The strategy includes adjusting the number and date formats from data that is extracted from the document so that the machine on which it is running can process the data correctly.

For example, many European countries use a comma as a decimal separator, but the US and UK use a period as a decimal separator. This practice is normally not a problem unless a machine is trying to process documents generated from locales other than its own. When this case occurs, data localization actions are necessary.

The data localization actions can be placed in two general categories:

- ▶ Actions that affect rules execution
- ▶ Actions that affect the captured data

10.11.1 Actions that affect rules execution

Many Taskmaster actions require parameters, and sometimes these actions require parameters that must be localized. An example is the SetSearchArea action when you define how much of an image to compare when fingerprint matching.

To accomplish this task, the IsLocalDecimalSeparator() action is used. The parameter that is supplied is tested against the decimal separator that is used by the localization settings of the machine. If you supply a comma, for example, the action returns *true* if the machine is set to use commas as the decimal separator in the localization settings. Figure 10-22 shows how the action is used.

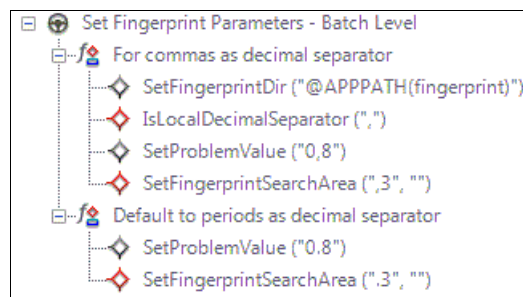


Figure 10-22 Rule using the IsLocalDecimalSeparator() action

When the localization settings of the machine are set to use a comma, the parameters supplied by the `SetProblemValue()` and `SetFingerprintSearchArea()` actions are supplied with commas. If the `IsLocalDecimalSeparator(",")` action returns *false*, the parameters are supplied with a period as the decimal separator.

10.11.2 Actions that affect the captured data

The data on the form can be a challenging to localize. A date value of 10/20/2011 can always be assumed to be 20 October 2011. However, a date value of 02/04/2011 can be 4 February 2011 or 2 April 2011 depending on the locale of the origin of the document.

Another example is in a field that contains a quantity. Some quantities are fractional. Therefore, it is difficult to know if 6.301 is referring to six thousand, three hundred and one or if it is referring to six and three hundred and one thousandths.

This type of ambiguity must be handled case-by-case, often by looking up the locale of the originator of the document and handling the data based on that locale.

For values such as currency, where you know that there is a decimal separator, the `CheckAndFixLocalDecimal()` action can be an effective way to convert the data programmatically. This action looks at the rightmost separator, converts it to match the local machine, and removes the thousands separators if present. The currency values then contain only a decimal separator. At export time, those separators can be changed again to match the specified decimal separator of the system to which you are exporting.

10.12 Intellocate

Intellocate is the technology that allows Taskmaster applications to learn. With DNA technology, documents that have never before been processed by the system are added to the fingerprint library. Locate rules are used to automatically find some of the data from these documents by using keyword searches or regular expressions. However, what cannot be automatically found can be identified and captured quickly and easily by a verify operator by using the Click'n'Key capability.

After this task is done, Intellocate saves the zones for the fingerprint. Then the next time a similar document is encountered, the fingerprint is matched, and all of the data is read by the zones.

Before the invention of Intellocate, IT personnel were often asked to set up each document type before a system processed them. Because these documents can come in at any time from various sources without warning, this task did not prove to be an attractive method of dealing with the documents. Others systems only used locate rules to locate data from a form, with no setup, but with limited results. If an unusual keyword was encountered, the document might never be able to be processed correctly.

Taskmaster, with Intellocate, developed a hybrid of the two techniques to provide fast, accurate learning of new documents that are introduced into the application. The added benefit is that this task can be accomplished easily by data entry operators. These operators best know the forms that they are dealing with as opposed to the typically more expensive IT staff.

As explained in 10.4, “DNA technology” on page 314, fingerprints are sometimes added to the system programmatically, but without the zonal information required to extract data zonally. The document goes through the verification process, and an operator clicks the image to populate any fields that were not located programmatically. In this case, the zones of the data are saved so that you can later write them to the Setup DCO or FPXML, depending on the system you selected. You can also classify the fingerprint by the criteria that you choose.

In IBM Taskmaster Accounts Payable Capture, the fingerprint is classified by the vendor who originates the documents. In Taskmaster Flex, the fingerprint is classified by the document type. This process is called *Intellocate*. It is done in the export process to ensure that the data required to classify the fingerprint is completed. Also all zones are captured in the runtime DCO, so that those zones can be saved to the appropriate location.

To classify the document, use the SetFingerprint(SmartParam) action. The Smart Parameter can refer to a field value or a variable from the page or a field. Alternatively, it can be a value that you specify as text. For example, the SetFingerprint(@P/Vendor) action classifies the current fingerprint in a classification with the name of the value that is currently in the Vendor field, which is a child of the page. If the classification value does not exist, it is created for you. Otherwise, the fingerprint is added to an existing fingerprint class.

To save the zones for the current fingerprint, use the WriteZonesFPX() action from the FPXML action library, or use the iloc_SaveZones() and iloc_SaveDetails() actions from the Intellocate action library.

10.13 Flex technology

In Flex technology, fields are given extra attributes as variables at run time so that they can be located and validated by simple generic rules that are applied to all flex-type fields. With Flex technology, you can create flex-type fields at run time, so that you do not have to explicitly set up all document types in the Setup DCO before processing.

With Taskmaster Flex, for example, the document classes and the fields that you want to capture for each document are stored in a database. They are configured by the Flex Configurator utility. In other projects, Flex technology has been used to read document definition databases that are used to print the documents, capture them using Flex technology, and store them in an image repository with a single source of the document definition.

Taskmaster Flex reads the document database at run time and applies attributes as variables in the runtime DCO. A typical field in the DCO might look similar to the DCO field shown in Figure 10-23.

F	Invoice_Number
	Text value -
	Char conf -
	TYPE : FlexField
	STATUS : 1
	label : Invoice Number
	Position : 0,0,0,0
	FlexDataType : AlphaNumeric
	FlexMinLength : 2
	FlexMaxLength : 20
	FlexPictureString : NNNNNN
	FlexRequired : Y
	FlexAllowOnly :
	FlexActions :
	FlexUseZone : Y
	FlexLocateAction : FlexKeywordListFind(InvNum)

Figure 10-23 A DCO field with flex attributes

These additional variables contain all of the information needed for special flex actions to locate and validate the field:

LocateFlexField() Uses the variable attributes on a field to programmatically locate the data on a document.

ValidateFlexField() Uses the variable attributes on a field to programmatically validate the data in a field.

10.14 Conclusion

This chapter described the dynamic technologies that can be useful in solving some of the most difficult challenges in data capture.

Chapter 11, “Technical walkthrough Accounts Payable Capture” on page 339, guides you through the IBM Taskmaster Accounts Payable Capture product and shows you how to use these technologies in the context of a fully featured invoicing solution.



Technical walkthrough Accounts Payable Capture

This chapter guides you through the dynamic IBM Taskmaster Accounts Payable Capture application. This application is called a *foundation application*, because it is used as a starting point for capturing complex machine printed forms with line items in a dynamic fashion. This chapter describes Accounts Payable Capture as an example of using the dynamic technologies introduced in Chapter 10, “Dynamic technologies” on page 305. Before you read this chapter, you must read Chapter 10 to become familiar with the technologies as they are use in Accounts Payable Capture.

This chapter includes the following sections:

- ▶ Introduction to Accounts Payable Capture
- ▶ How IBM Taskmaster Accounts Payable Capture works
- ▶ Jobs available in the workflow
- ▶ When each task profile gets executed
- ▶ A walkthrough of the task profiles

Accounts Payable Capture versus APT: This book uses the name *Accounts Payable Capture* to refer to what was previously known as *APT*. Keep in mind that some windows in the application and window captures in the book might still use the name APT.

11.1 Introduction to Accounts Payable Capture

The IBM Taskmaster Accounts Payable Capture application is a learning workflow. It uses dynamic technologies to learn new instances of the documents that are introduced in the system. The Accounts Payable Capture workflow has been used for many types of documents and applications, not just invoices.

In this chapter, you look at the role that IBM Taskmaster Accounts Payable Capture plays in the Accounts Payable (AP) industry. You are introduced to the different jobs in the workflow and examine the task profiles, rule sets, rules, functions, and actions that make it work.

Because Accounts Payable Capture is constantly evolving with new technologies and techniques, your version might differ from what is shown in this chapter. However most of the techniques are applicable to any version. After you have a thorough understanding of how the technologies interact, you will be in a better position to apply them to any product where you need similar capability.

11.2 How IBM Taskmaster Accounts Payable Capture works

IBM Taskmaster Accounts Payable Capture captures the data from invoices and validates that data with business rules and with the business applications of the customer. Then it exports the image to an image repository and the data to the business applications.

In a typical application, a purchase order (PO) is entered into a PO system. After approval, the PO is sent to a vendor for fulfillment. The vendor ships the goods and a shipping document.

Then the customer receives the goods and enters the shipping document into the business application as a Proof of Delivery (PoD). Next, the vendor sends an invoice, which is received by the Accounts Payable department. The AP department uses IBM Taskmaster Accounts Payable Capture to capture the invoice. Accounts Payable Capture interacts with the business application system to obtain the most accurate data possible.

Next, the data is inserted into the business application where a three-way match occurs. In this process, the PO line items, the PoD line items, and the invoice line items are compared. When they match and the customer is confident that they ordered and received the goods, the business application system issues a check to the vendor.

Figure 11-1 illustrates this process in a simple diagram.

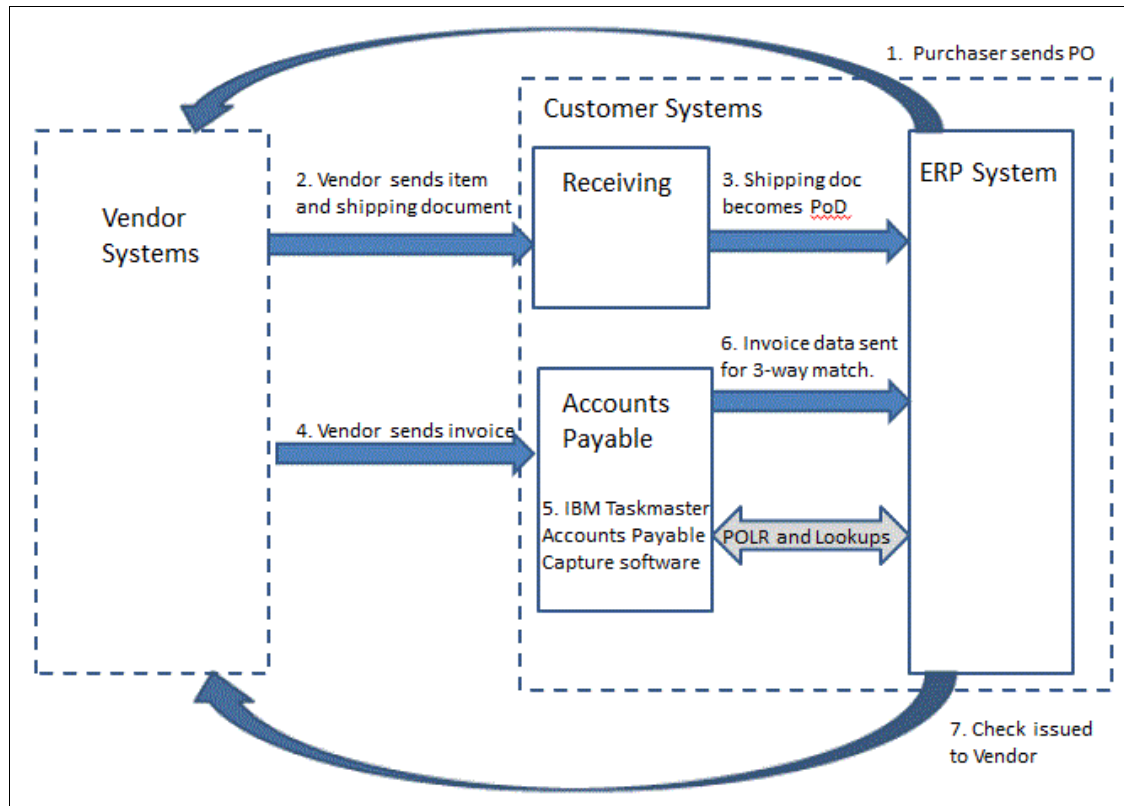


Figure 11-1 A diagram of the AP process

11.3 Jobs available in the workflow

You might notice many different jobs in the Accounts Payable Capture workflow. Although all of these jobs share common elements, they are different in some way.

Figure 11-2 shows a list of the jobs that are available in a current version of IBM Taskmaster Accounts Payable Capture.

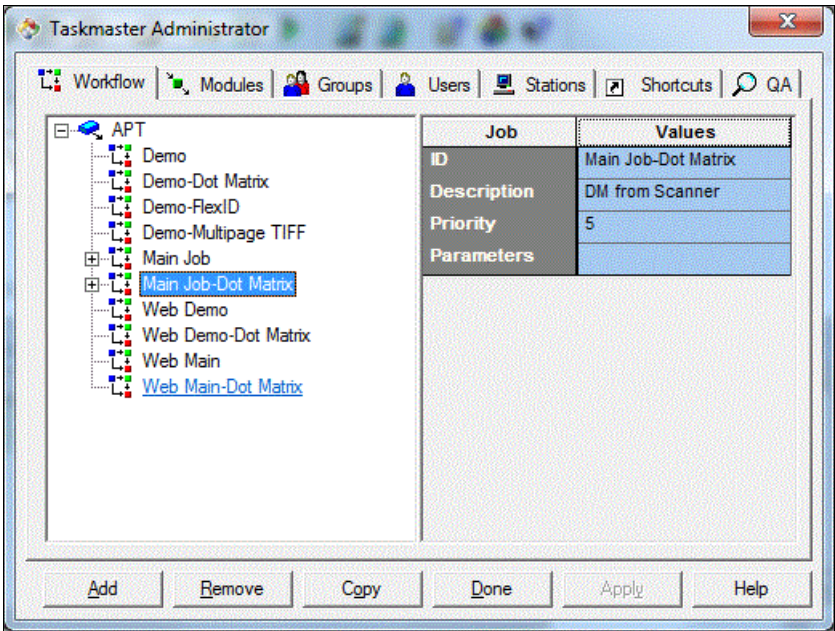


Figure 11-2 Many different jobs in Accounts Payable Capture workflow

Jobs that start with the term *Web* use web scanning and upload, but the tasks that follow are identical to their thick-client workflow counterparts.

Jobs listed as *Demo* use a virtual scanning technique, but jobs listed as *Main* are used to drive a physical scanner. You cannot set up the thick client tasks without a scanner and drivers attached to the machine. By default, the IScan task and module are not present. The jobs begin with Batch Profiler, but are intended for you to add a physical Scan task as the batch creation task and place it as the first task in the job.

The job or jobs that end with *FlexID* have an optional FlexID task before Batch Profiler to manually identify key pages of the batch that are necessary for page identification. For more information, see 10.3, “FlexID” on page 313.

The jobs that end with *-Dot Matrix* are identical to their counterparts in all respects except for the name of the job. At run time, during the recognition phase, the current job name is queried. If the job ends with *-Dot Matrix*, the recognition engine is programmatically configured for dot-matrix recognition through the use of rules.

The job or jobs that end with *-Multipage TIFF* are also identical to their counterpart-based jobs, except for the name. The current job name is queried during run time to programmatically enable rules that will do page identification based on the structure of the multipage TIFF files that were used for input.

This technique of using identical jobs with different names is useful in applications you develop where minor changes are in the rules execution that you want to enable or disable by placing the images into different jobs.

11.4 When each task profile gets executed

Tasks generally run task profiles, but APT (now Accounts Payable Capture) pioneered the use of running task profiles on demand from a verify panel. This process allows far more power in the Verify tasks without scripting them in a significant way. By using rules, which you run by clicking a button in a Verify panel, you can reuse routines multiple places in the execution of your application. You can also have a single-source code base for much of your application. In general, calling rules from a Verify panel is superior to coding and maintaining the same functionality in the panels themselves.

Figure 11-3 shows the task profiles uses within a typical Accounts Payable Capture workflow.

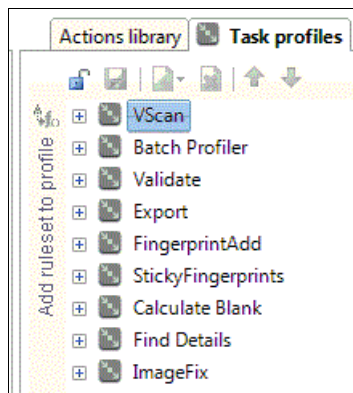


Figure 11-3 Task profiles in Accounts Payable Capture

The first four task profiles are run by tasks. Except for the Validate task, the task profiles are named the same as the tasks that call them. The Validate task is called to validate your document from a data entry panel during the Verify task.

FingerprintAdd is called when you add a fingerprint manually in the **DStudio Zones** tab. With an Accounts Payable Capture workflow, you do not do this task often, but it is possible. Accounts Payable Capture workflows can add fingerprints to the system automatically and with greater accuracy to the line item zones than is possible with the manual interface.

StickyFingerprints (see 10.5, “Sticky fingerprints” on page 317) is called from the Verify panels when the Verify panel detects a <New> fingerprint that it has learned zones for, from a previous image in the batch.

CalculateBlank and FindDetails run from buttons on the various Verify panels.

ImageFix is called when setting up the image settings on the **Zones** tab of Datacap Studio.

11.5 A walkthrough of the task profiles

The remainder of this chapter takes you through the task profiles of the Accounts Payable Capture workflow, as used in IBM Taskmaster Accounts Payable Capture at the time of this writing. This walkthrough is provided as an example of how to use the dynamic technologies in an application as explained in Chapter 10, “Dynamic technologies” on page 305.

Before you attempt the walkthrough, you must understand the concepts of rule sets, rules, functions, actions; how they are attached to the Document Hierarchy (DCO); and how they are run. For information about these concepts, read Chapter 10, “Dynamic technologies” on page 305.

This section highlights the following task profiles:

- ▶ The VScan Task Profile
- ▶ The Batch Profiler Task Profile
- ▶ The Verification process
- ▶ The Export Task Profile

11.5.1 The VScan Task Profile

The VScan Task Profile brings documents into the system programmatically and without user intervention. Other methods are possible for bringing documents into the system in an Accounts Payable Capture workflow. In these methods, the jobs listed as Main are generally devoted to driving a scanner. Also with the Web demo VScan task, users can choose the images that are to be used as input from a directory listing.

Other Accounts Payable Capture jobs that are common, but not in the foundation application at the time of this writing, are tasks that pull documents from an email system or a fax server.

Tweaking your application by using the VScan rule set: Set up, develop, and tweak your applications by using the VScan rule set. When tweaking the system for optimal performance, you will often find it advantageous to run the same images over again so that you can tell if your modifications have improved the process. This approach is preferred to running a subsequent batch that is scanned a little straighter.

The VScan Task Profile runs in the demo jobs that are not designated as Web in the Accounts Payable Capture workflow (Figure 11-4).

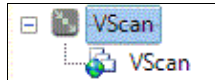


Figure 11-4 The VScan Task Profile

By default, the VScan Task Profile only contains one rule set, which is also called VScan. It is common to add another rule set to this task profile if your input images are PDF, JPEG, Grayscale, Color, or another image type that must be converted. It is advantageous to have all images that are finishing this task profile as bi-tonal TIFF images. If you place the convert actions first in the Batch Profiler Task Profile, rolling back batches to the start of Batch Profiler becomes an issue. The images try to go through the conversion process again.

The VScan rule set

The VScan rule set is typical for a system that has single-page or multipage TIFF files in the input directory (Figure 11-5).

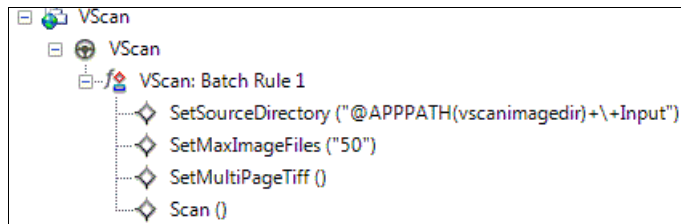


Figure 11-5 The VScan rule set

As shown in Figure 11-5, SetSourceDirectory indicates that VScan must look for images in an Input subfolder off the folder that was specified in the vscanimagedir tag of Application Service. It is preset to accept a maximum of 50

images and to burst multipage TIFF files into single-page TIFF files if they are pulled in.

11.5.2 The Batch Profiler Task Profile

The Batch Profiler Task Profile is the main background processing profile. This task profile is used in every job in the Accounts Payable Capture workflow and performs all processing between image acquisition into the system and the verify process.

A few of the rule sets, such as Purchase Order Line Reconciliation (POLR) and VendorNumberLookup, are specific to invoice processing. They might not be needed in other learning applications.

Figure 11-6 shows the rule set composition of the Batch Profiler Task Profile.

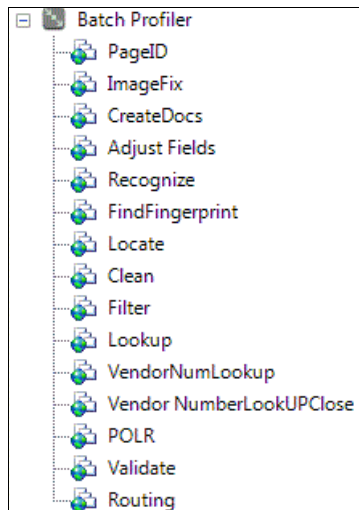


Figure 11-6 The Batch Profiler Task Profile

The PageID rule set

Accounts Payable Capture is the first workflow to use the Advanced PageID actions that are described in 10.2, “PageID actions and techniques in dynamic applications” on page 307. These actions are included in PageID.rrx file that is in the rules directory of Accounts Payable Capture. These actions can be copied to any application that needs this capability.

Figure 11-7 shows that the implementation of these actions is in the PageID rule set.

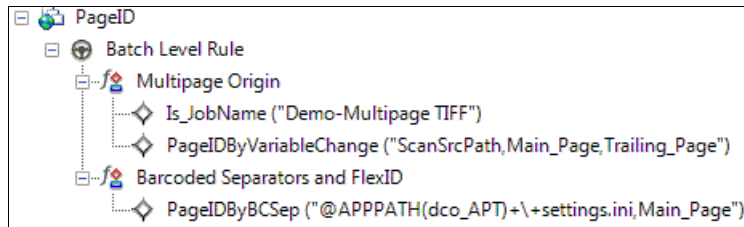


Figure 11-7 The PageID rule set

Only one rule is in this rule set, and it is attached at the batch level. When the DCO enters this rule set, every image in the batch has only a batch-level object with a child page of type *Other*. When VScan runs, it places the ScanSrcPath variable on each page that lists the source TIFF that the image came from. In a case where VScan breaks out images from a multipage TIFF or PDF file, several images in a row can have the same ScanSrcPath variable.

In the first Multipage Origin() function, the first action, the Is_JobName() action, checks which job name you chose at scan time. If the action checking the job name returns *true*, the PageIDByVariableChange() action runs, querying the ScanSrcPath on each image and naming the pages.

When the ScanSrcPath changes from one image to the next, the image with the new ScanSrcPath value is named according to the second parameter of the PageIDByVariableChange() action. In this case, it sets such page type as Main_Page. For every subsequent page in the batch that contains the same value in ScanSrcPath, the images get the type specified in the third parameter, which is Trailing_Page in this case.

If you are not running the demo Multipage TIFF job, the first action, the Multipage Origin() action, returns false, and it runs the PageIDByBCSep() action. This action reads the settings.ini file and uses that information to set the types on all the pages. As mentioned in 10.2.1, “Page identification by barcode separator” on page 310, you only need barcode separators in multipaged documents in the batch. Single-paged documents are placed at the start of each batch and get the type specified in the second parameter, which is Main_Page in this case.

After PageID() runs, all pages must be named. If a page is not typed as *Other* going into the PageID, it is not renamed. The reason is that FlexID might have named the images before PageID runs, and PageID must not overwrite what an operator has specified as the type. However, if FlexID has run, the actions set the

type on any Other page that is not renamed in FlexID, according to the LastPage_ThisPage section in the settings.ini file.

The ImageFix rule set

When ImageFix is used in the Accounts Payable Capture workflow, it follows PageID. With form-type applications, where fingerprinting is often used for the page identification, ImageFix runs on every page and before the fingerprint match is attempted.

With this type of workflow, with PageID before ImageFix, you can be more aggressive with the line removal process because the barcodes will have been read and processed. However, particularly for invoices, you need to be less aggressive on the despeckling process, because you need to keep the decimal points in the currency fields if possible.

ImageFix in the Accounts Payable Capture workflow also automatically rotates the images that it processes before the settings in imagefix.ini file are applied to the image (Figure 11-8).

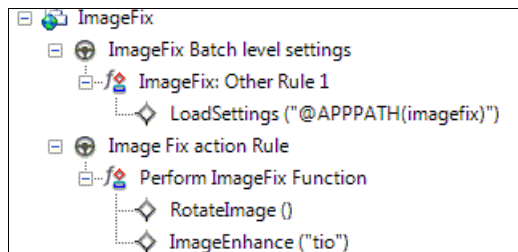


Figure 11-8 The ImageFix rule set

RotateImage() creates a CCO file to do its work, unless one was already created. It uses the AnalyzeImage method of CCO creation, which uses the location of graphics, words, and lines. The CCO file is created before the image is deskewed. Therefore, you do not want to use this CCO file for fingerprint matching. You can create another one during the Recognize rule set.

The CreateDocs rule set

The CreateDocs rule set in an Accounts Payable Capture workflow is the same as it is in all workflows. At the batch level, you run the CreateDocs rule set, and at the page level on pages that contain fields, you run CreateFields.

Figure 11-9 on page 349 shows the variables that the CreateDocs rule set uses to create a document structure in the Accounts Payable Capture workflow. When using the CreateDocs rule set on the Main_Page object, it always sets it as the first page of a new document called the *Invoice document*. The Invoice document

can contain all of the other page types that are set in PageID, except for a Document_Separator page. Those pages are placed in a Separator document when found. No additional processing is done on Separator documents in Accounts Payable Capture. Therefore, the process effectively culls out the Document_Separator images from the batch, without removing them, leaving your audit trail intact.

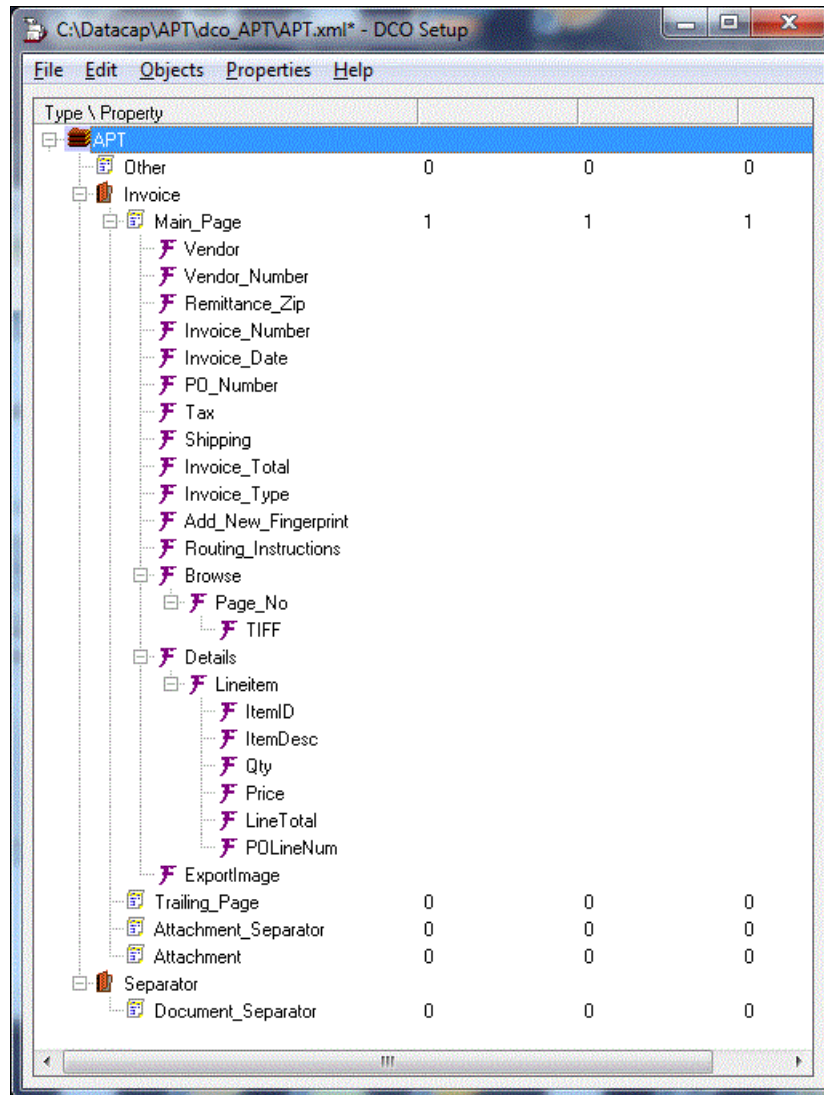


Figure 11-9 Accounts Payable Capture setup DCO variables for Min, Max, and Order

The AdjustFields rule set

The AdjustFields rule set creates a document structure called *Browse* so that users can have a structure that spans each page of the invoice, and you can browse between them when using Taskmaster Web. The AdjustFields rule set creates a line-item child, called *PageNo*. PageNo has a subfield named TIFF that gets positions in the upper-left corner of each page and a value set to the current TIFF name that the field is placed on.

When you are in the Verify panel and you click a field, it automatically shows the page on which the field is found. If you need to move to a page to click data, the thin client does not have buttons to make such a move. Therefore, this browse structure places a field on every page that you can use to effectively move back and forth between pages of the invoice.

Figure 11-10 shows the implementation of the AdjustFields rule set. The Add Page_NoReferences runs on the Browse field on Main_Page.

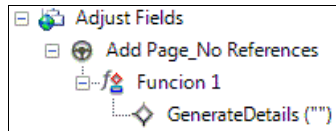


Figure 11-10 The AdjustFields implementation

The Recognize rule set

In the Recognize rule set, some Job Specific actions occur. This rule set also provides an interesting technique for capturing statistics in the runtime DCO. For more information about, managed recognition, see 10.6, “Managed recognition” on page 321.

Figure 11-11 contains the rules, functions, and actions of the Recognize rule set used in Accounts Payable Capture. The single page-level rule is attached to the Main_Page and Trailing_Page objects. The Attachment pages are not recognized.

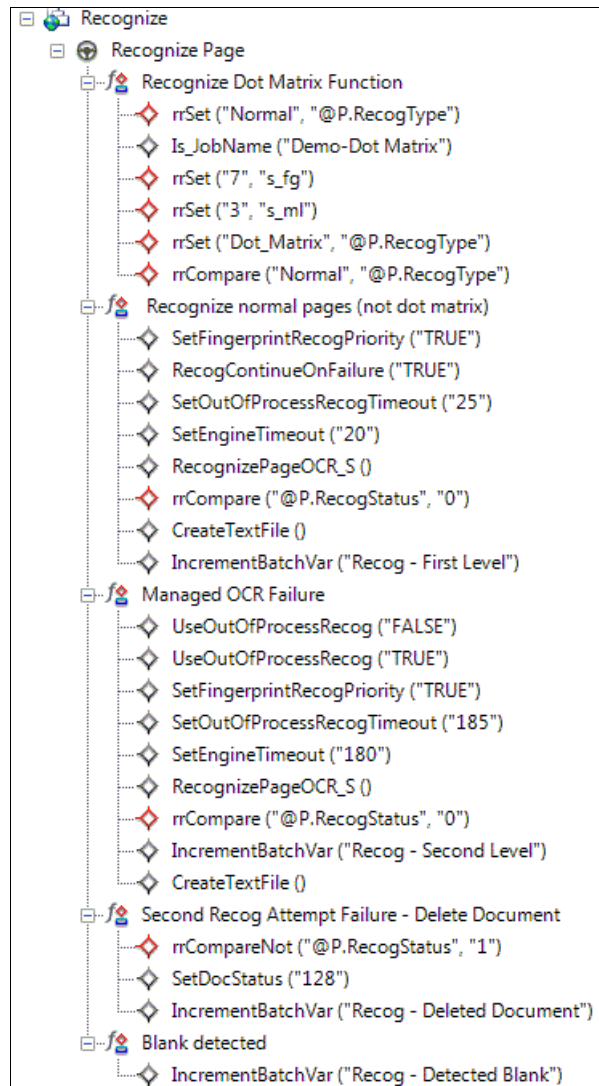


Figure 11-11 The Recognize rule set

The first function, the Recognize Dot Matrix function, always returns *False*. The first action, which is the `rrSet()` action, creates a batch-level variable called *RecogType*. The action sets its value to *Normal*. The second action, which is the `Is_JobName()` action, then checks the job name, which is Demo Dot Matrix in this

case. If this name is not the current job name, the action fails, and the next function is immediately started. If the value is *True*, the action adds variables to the page and sets the value of those variables to inform the ScanSoft recognition engine later to use the dot-matrix settings. It then sets the batch-level *RecogType* variable to a value of *Dot Matrix*, and the last action, which is the *rrCompare()* action, then returns a value of *False*.

You might need to copy this function if you have more than one job that you want to use for Dot Matrix. You need a function before the Recognize Normal Pages (not dot matrix) function for every job that you add for Dot Matrix recognition. For example, if you want to add a Main Dot Matrix job, you copy the top function and place the copy above the Recognize Normal Pages function. Then you change the *IsJobName* parameter to the name of your new job.

The second function, which is the Recognize Normal Pages function in Figure 11-11 on page 351, shows the first attempt at managed recognition. It is set with the shorter timeouts.

It begins with the *SetFingerprintRecogPriority()* action set to *TRUE*. By setting this action, *RecognizePageOCR_S* replaces any existing CCO with a new CCO that is created by the recognition process. This action is necessary because *Rotatelmage* was used in the *ImageFix* rule set. It creates a CCO with the *AnalyzeImage* method and was created before the image was deskewed. This action essentially throws away the CCO and makes a new CCO based on where words and lines were recognized by the recognition engine. This action provides a more usable CCO for locating data than a CCO that was created with *AnalyzeImage*.

The *RecogContinueOnFailue()* action is then set to *TRUE*. This way, if a problem image is in the batch, the batch can continue after you detect this problem and reset the engine.

The next two actions, which are the *SetOutOfProcessRecogTimeout()* and *SetEngineTimeout()* actions, specify the timeouts to use. Figure 10-11 on page 322 shows an example of managed recognition. A monitored thread is created that has been set to automatically shut down in 25 seconds if recognition is unsuccessful. The recognition engine itself is created in that second thread and has its internal time-out set to 20 seconds. In most cases, a page is recognized quickly, usually 2 seconds or less. However, if a problem image is encountered, the recognition engine can detect this image by the time it takes to recognize it. Failing that test, the thread expires if the recognition engine hangs and is unable to monitor itself.

The *RecognizePageOCR_S()* action does the recognition. It uses variables that are created with the engine setup on the **Zones** tab and perhaps the Dot Matrix variables. It tries to recognize the page, write the CCO, and return a status.

If the status that it returns is 0, everything is successful. A text file is then created in that batch directory with the `createTextFile()` action. This text file is for observation and troubleshooting only. It is not used elsewhere in the process.

Finally, a batch-level variable is created or incremented with a special action called the *IncrementBatchVariable()* action. This action helps to capture statistics and place them at the top of the runtime page file so that you can quickly look at the page file. This technique is useful if you have several branches that your images can be processed with because it counts the number of times that a path is taken. If the variable does not exist, it is created with a value of 1. If the variable does exist, the value is incremented by 1.

The Managed OCR Failure function is a copy of the function before it tries recognition a second time with longer timeouts, but after resetting the engine. The `IncrementBatchVar()` action logs in the DCO when the managed failure occurs.

If the engine fails a second time, the Second Recognition Failure function deletes the document. However, if the recognition engine returns a value of 0 for `RecogStatus`, the Managed OCR Failure completes. If the recognition engine returns a value of 1 for `RecogStatus`, the recognition engine detects a blank page.

If the page is not blank, we have tried to recognize it twice, once with a 20 second timeout, and another time with a 3 minute timeout. In this case, mark the page for deletion, which also deletes the document it is in. This action does not delete the document. Instead, it marks the document so that you do not attempt to process it further. A well-designed system notifies someone about these deleted documents. Accounts Payable Capture does this notification in the Export task.

The FindFingerprint rule set

Fingerprinting on the Accounts Payable Capture workflow is only done on the `Main_Page`. The purpose of the Fingerprint rule set is for automatic identification of the specific vendor who sent the invoice. It also provides the offset that is needed to apply to zones to compensate for differences in the scanning positions.

If the Accounts Payable Capture software is being used to capture documents from tens of thousands of vendors, use the Fingerprint Service. The Fingerprint Service is a web server that stores the CCOs for all active fingerprints all the time. Without it, a background computer must load the CCOs from the network every time a batch is processed, which can add significant time to the background processing.

The role of the FindFingerprint rule set is to ensure that every `Main_Page` has a fingerprint `TemplateID` that you are working with. If no sufficient match is found with an existing fingerprint, a new fingerprint is created automatically. Any new fingerprints created this way do not have zones defined on them. Also, all data

extracted must be by keyword or regular expression before a data entry operator views and processes the invoice. Later in the “The PreExport rule set” on page 379, we run Intellocate rules that save any zones that are identified for these new fingerprints. This way, the next time that this invoice is encountered, the data can be found by using zones.

Figure 11-12 shows the FindFingerprint rule set that is used in an Accounts Payable Capture workflow.

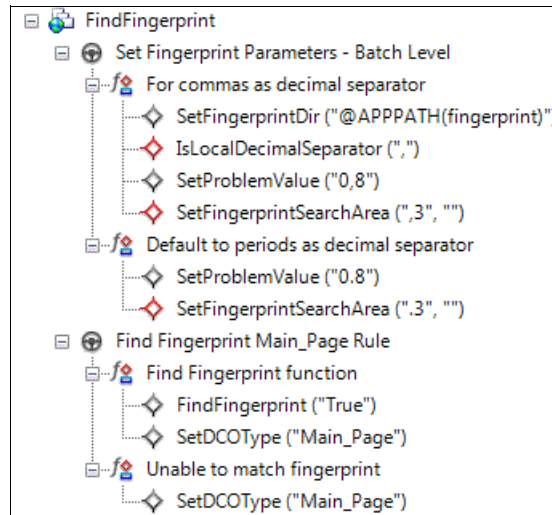


Figure 11-12 The FindFingerprint rule set

The batch-level rule sets the fingerprint directory from the App Service setting. The new fingerprints are stored in this directory if they are created. The paths to the existing fingerprints are in the Fingerprint database.

Moving the Fingerprint database from one system to another: With a learning system, such as Accounts Payable Capture, use care when moving the Fingerprint database from one system to another. You must ensure that the paths are correct and that you do not lose entries when moving.

Consider what might happen if you copy your entire system from production to development, work on the system in development for a few days, and then try to copy the entire system back. You might lose any new fingerprints that were added to the production system while the copy was in development. In general, do not move the fingerprint database from one system to another.

From the batch-level rule in Figure 11-12, the first function, which is called *For commas as a decimal separator*, checks the locale of the machine that is

processing the fingerprints. It ensures that the floating point values from SetProblemValue and SetFingerprintSearchArea use the correct decimal separation character in their parameters.

Normally, we want to look at the top 30% or so of the current image to compare it against our fingerprint library for matching. The ProblemValue, by default, is set to 0.8. Decreasing this value increases the chances that the fingerprint mismatches a fingerprint from another vendor, creating fewer fingerprints in the system overall. Increasing this value gives a more precise match, creating creates more fingerprints in the system and additional work for data entry operators in zoning these additional new fingerprints.

Adjust this setting only after a discussion with the users about the effects that they can expect by changing this value. Often times, we find that a value of 0.8 provides the right balance.

If a mismatch occurs, the data entry operators can create a fingerprint with the click of a button, or in the case of the web, by choosing **YES** from the Add New Fingerprint list. If more than one fingerprint exceeds the ProblemValue setting, the fingerprint with the best match is returned by the FindFingerprint action.

The SetDCOType action ensures that the FindFingerprint action does not change the page type (which is set in the PageID rule set) to another value.

The Locate rule set

In the Locate rule set, the data is pulled from the CCO that was created in the Recognize rule set into the fields of the DCO. The rule set is extensive, but overall employs a few different techniques for retrieving the values. We explore each technique.

The Locate rule set also runs as part of two different Task Profiles, Batch Profiler and StickyFingerprints. Figure 11-13 shows how the Locate, Clean, Filter, and Validate rule sets are used during the StickyFingerprint Task Profile, which is called during the Verify process.

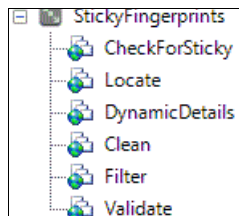


Figure 11-13 The StickyFingerprints Task Profile

Reusing rule sets this way reduces the maintenance of the system. However, it makes each rule set slightly more complex. Each rule set must test when it is being run so that it knows precisely how to handle its order of operations when processing or reprocessing a document. If you are unfamiliar with what StickyFingerprints are, see 10.5, “Sticky fingerprints” on page 317.

Figure 11-14 shows the document- and page-level rules of the Locate rule set.

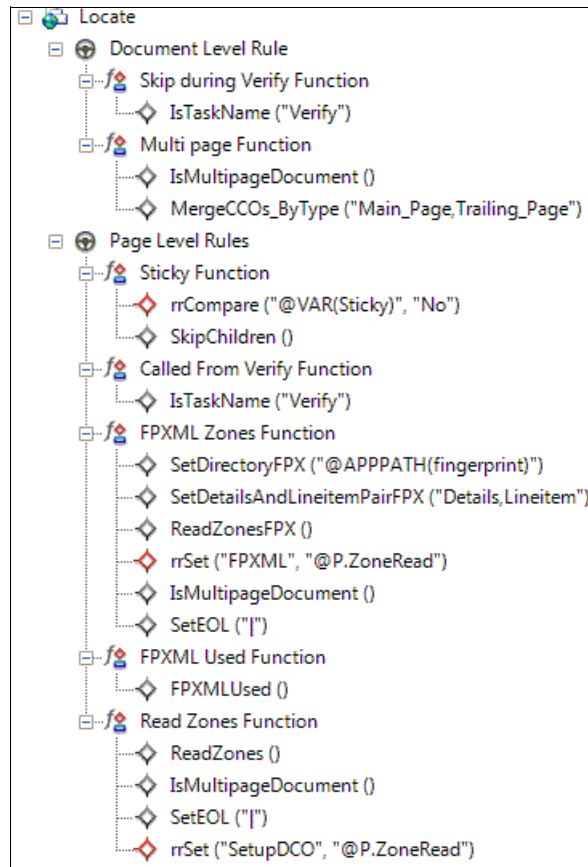


Figure 11-14 Document and page-level rules in the Locate rule set

The document-level rule is in place to make a multi-CCO file (MCCO) that combines all of the CCOs from the Main_Page and any Trailing_Pages into one large CCO. This way, the entire document can be searched for data at one time. For more information about this technology, see 10.7, “CCO Merging” on page 323.

When making an MCCO, only do it once and only do it during the Batch Profiler Task Profile. For this reason, the check is in place on the document-level rule to

see if the document is being reprocessed during the Verify Task. In the document-level rule, you can see that, if the batch is being reprocessed by Verify, it does not attempt to remake the CCO file. The MCCO technology adds the Trailing_Page CCO to the Main_Page CCO and replaces the CCO of the Main_Page. Therefore, running this action more than once has the effect of duplicating the data from the Trailing_Pages to the CCO and potentially doubling the data found when we search it.

In the first function on the document level, we check to see if we are running under the Verify Task. If we are, the function returns *true* and does not run the second function that creates the MCCO.

In the second function, the `IsMultipageDocument()` action determines if the document contains more than one page. If it does, it makes an MCCO for you.

For the page-level rule, we are trying to read any zones that are associated with the fingerprint if they exist. Similar to the document-level rule, we must check to see when the Locate rule set is running so that we not to reprocess documents that do not need it.

The first function checks to see if we are in a situation where StickyFingerprint technology is useful. See the StickyFingerprint Task Profile listed at the top of this section (see Figure 11-13 on page 355). The first rule set in the StickyFingerprint Task Profile sets a variable named *Sticky* that indicates whether we need to reprocess the runtime document with Locate rules. If the value of the Sticky variable is No, the `SkipChildren()` action causes the rule set to stop running for all children of the page, meaning that the fields will not be searched for data.

Regardless of whether Sticky is set to Yes or No, we do not want to attempt to read the zonal information when we run from the Verify Task. The reason is that the first rule set in the StickyFingerprint Task Profile also sets the zones based on the previous document in the batch. The net result of the first two rule sets is that, when running under StickyFingerprints, the zonal data is never read by the Locate rule set. Depending on whether the Sticky variable is set to Yes or No, the field-level actions in the Locate rule set will or will not run.

The remaining functions on the page-level rule must only run during the Batch Profiler Task Profile. Again, because there are two different places where zones can be stored (FPXML or the Setup DCO), we must check to see which actions are appropriate for reading the zonal information.

Reading zones with the FPXML method requires us to set the fingerprint directory where the FPXML files are kept. To read FPXML with detail lines defined, we must inform the `ReadZonesFPX()` action of a detail structure so that it can read and apply the detail line zones correctly. This process is done by

using the `SetDetailandLineItemPairFPX()` action. You do not have to specify the Browse structure here. Such zones are previously set programmatically in the `AdjustFields` rule set.

`FPXReadZones` returns `FALSE` if it cannot find or read an `FPXML` file. If it is a multipaged document, set an EOL character for the `MCCO` to process correctly. If the `FPXML` is read, then a variable is set at the page level so that we know the method by which the zones are read. Because there are two actions that can return `FALSE` in this function, we must check this variable in the next function to see if the zones are read in correctly. If the `ReadZonesFPX()` action returns `FALSE`, the variable is not set, and it attempts to read zones from the `Setup DCO`.

The rest of the `Locate` rule set pulls data into the fields by using the following methods:

- ▶ Populating data that is normally best found zonally
- ▶ Finding data that floats around
- ▶ Searching for and populating the detail lines

Populating data that is normally best found zonally

With the first two methods, `Accounts Payable Capture` checks zonally and with keyword searches. The difference is the order of the methods that are used. Figure 11-15 on page 359 shows how we find data when zones are preferable for locating the data. We use this method when data does not tend to float around the form for these field types.

As with the previous rules in this rule set, we must check which task we are running under. Therefore, the first function checks to see if we are running under the `Verify` task. If we are, then it loads the `CCO` dynamically and attempts to populate the data from a zone with `PopulateZNField()`. If we are not running under the `Verify` task, the `CCO` is already loaded, and we can immediately run the `PopulateZNField()` action. If a zone is defined for the field and the zone contains data, the data in that position is pulled into the field. The rule for that field is finished, and we can move on to the next field.

If no zone is defined (as in a new fingerprint) or if no data is in the zone, we use `Locate` actions to try to find the data programmatically.

The first step in this process is to find a keyword or regular expression. Although regular expressions are a preferable method of finding data in general, most of what we capture from invoices is not unique enough in structure to justify finding by using a regular expression. The exception is a `PO` field. If a company uses a number that is fairly unique, such as the letters `PO` followed by 8 digits or something of that nature, consider replacing the default actions to find the `PO` by regular expression.

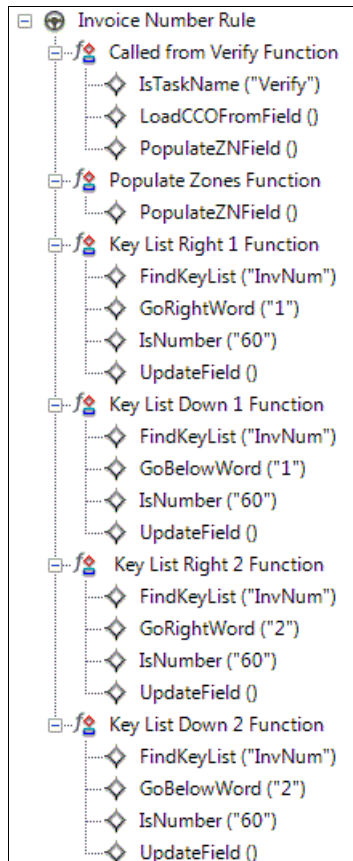


Figure 11-15 Finding data with the zonal method preferred

With the keyword search, a key file is placed in the dco_<AppName> directory listing the keywords that might be used as labels around the data that you are searching for. Accounts Payable Capture ships with a limited set of keywords in the .key files. Add additional keywords as you find appropriate.

After a keyword is found, we move one word to the right to see if that word contains data that is appropriate to the type of data we are looking for. If it does, we do an UpdateField action, and then we are done. The rest of the functions in the rule operate identically, but look at different words based on their location related to the keyword that was found.

This method is used for the bulk of the fields in a document where the data for that instance of a document is located in a static location (because we are checking for the zone first).

Finding data that floats around

Some data, such as the Invoice Total, Tax, and Shipping, are normally found on the last page of the document. Invoices have a variable number of pages. One day you might get an invoice from a vendor that has one or two line items and everything fits on a single page. The next day, you might get an invoice from the same vendor that has dozens or hundreds of line items and is many pages long.

For this reason, we use a different preferred method for data that typically falls below the last line item on an invoice (Figure 11-16).

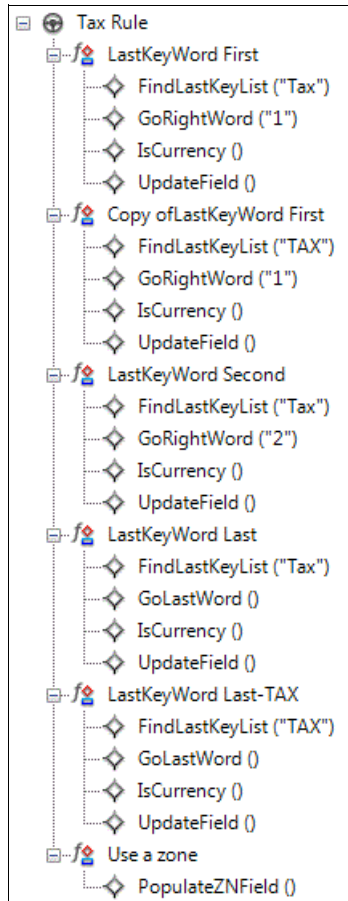


Figure 11-16 Rule for finding data that tends to float

In this case, we look for the last occurrence of a word in a document and then look around that word for data that fits the data type we are searching for. If the word is not found, then we use the zone as a last resort.

Searching for and populating the detail lines

The technology involved in finding detail lines is explored in 10.9, “Line item detection” on page 325. However, more is involved in the actual implementation than just the actions that search the CCO for lines as illustrated in Figure 11-17.

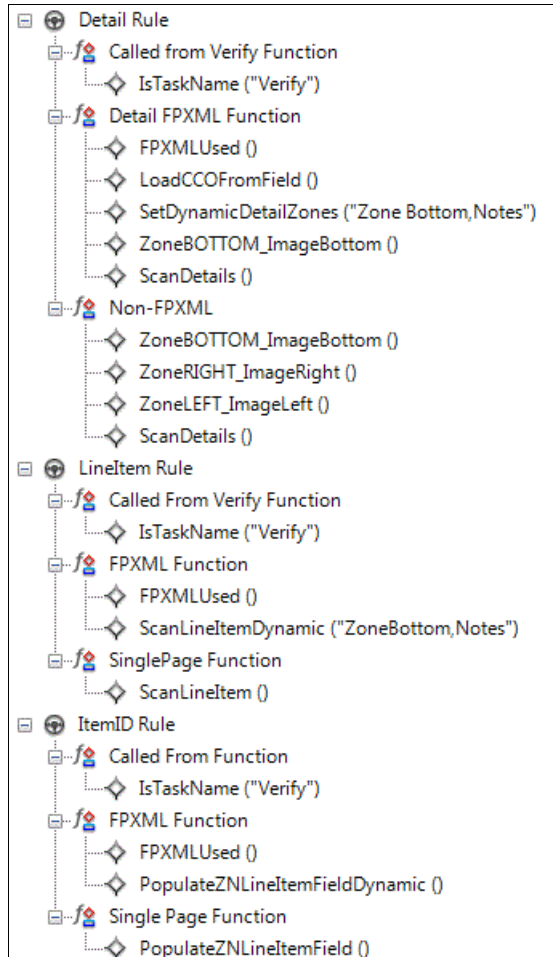


Figure 11-17 Finding detail lines in the Locate rule set

As with some of the other rules in the Locate rule set, we must check if we are running in the Verify task to prevent reprocessing the document unnecessarily. The first function does this check. If you are running under the Verify task, the rule set does not use these actions to find detail lines.

The second function checks whether we are using FPXML and moves the FPXML zones to the detail structure. FPXML is different in its storage of zones

than in the storage of the Setup DCO. If FPXML is used, we must load the CCO again. The next two actions set up the zone for the multipaged document, ensuring that the entire CCO is searched for line items, even though it contains a variable number of pages. Regardless of whether FPXML is used, after the zone is set up, ScanDetails creates the detail structure and puts each line item in its own child Lineitem field.

At the line-item level, we perform a ScanLineItemDynamic action, by using the recently loaded CCO if FPXML is used. Alternatively, we perform the ScanLineItem function if the zones were read from the Setup DCO.

Finally, each field on the line item is populated with PopulateZNLItemField.

If you are unfamiliar with how this technology works, see 10.9, “Line item detection” on page 325.

The Clean rule set

The Clean rule set is used for a specific purpose on a limited number of fields in an Accounts Payable Capture workflow. Most field cleaning occurs in the Validate rule set. When a data entry operator clicks a field, we want to remove unwanted characters and set the data format in fields, such as dates, to a specified format.

However, some fields are used for data lookup before we validate them. Therefore, those fields must be cleaned and potentially formatted in this rule set. Also, we must check the data in certain line-item fields to ensure that it is the data that we are searching for before filtering the line items. If you are unfamiliar with this method, see 10.9.4, “Filtering line items” on page 330.

Figure 11-18 shows the default rule set for cleaning invoices.

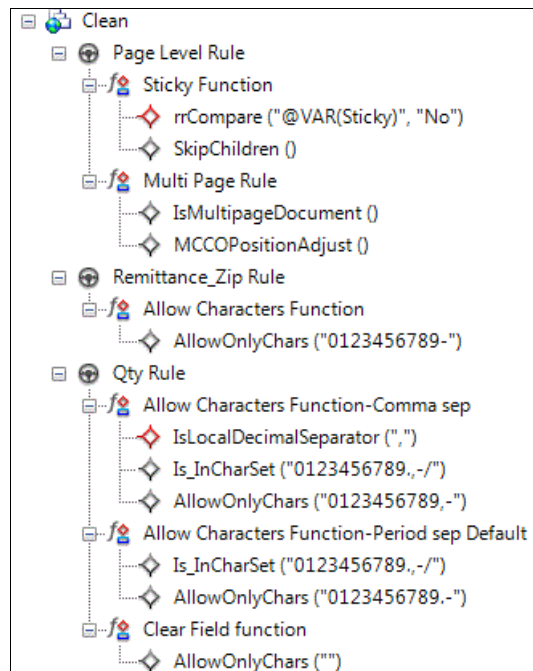


Figure 11-18 The Clean rule set

The page-level rule indicates that the Clean rule set must do nothing if we are running under the StickyFingerprints rule set and the current fingerprint is not eligible for sticky processing. If the rule fails, then a first attempt is made to adjust the field positions from the MCCO to coordinates that are associated with each page. For more information, see Chapter 10, “Dynamic technologies” on page 305.

Because the ZIP field is used in a vendor lookup, the data is cleaned of any character that is not a number (or potentially a dash, if postal codes contain dashes in your database). This action ensures that we have removed any spaces or extraneous characters that might cause the lookup to fail.

In the line-item fields, we delete any unexpected data from those fields. For example, if a Qty field contains the word “Thank,” which it might have captured when reading the bottom of an invoice, this word is deleted. The same is done for Price and Line Total in the detail-level fields. For more descriptions about why this is done, see Chapter 10, “Dynamic technologies” on page 305.

Avoid the urge to clean fields in this rule set that are not used in lookups or for detailed line filtering. Otherwise, you have to do it again in the Validate rule set, which adds to maintenance if the cleaning parameters change.

The Filter rule set

The Filter rule set is used to delete the line items that are captured during the Locate or FindDetails rule sets that do not fit the data types we are expecting. For more information about this technology, see 10.9.4, “Filtering line items” on page 330.

The Filter rule set used for invoices in the standard Accounts Payable Capture workflow is shown in Figure 11-19.

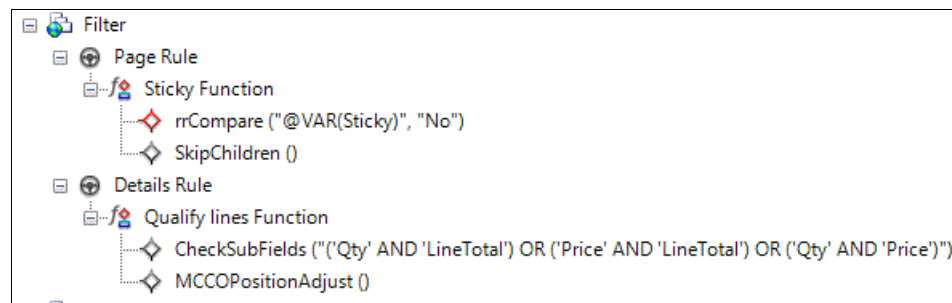


Figure 11-19 The Filter rule set

Again, the page-level rule avoids reprocessing the line items if we are not in a condition where it is beneficial to do so when running under the StickyFingerprints Task Profile.

In the detail field, the CheckSubFields erases all line items that do not have valid data (after cleaning) in two of the three Qty, Linetotal, and Price fields. After cleaning, MCCOPositionAdjust adjusts the raw CCO coordinates for a multipage CCO into coordinates that are adjusted to the single page image from which the data originated.

The Lookup rule set

The lookup rule set populates the Vendor name in the vendor field. The Vendor name is stored in the fingerprint database. However, the rule set has an option for pulling the information from a locally stored database. An example is a mobile computer for development that might be disconnected from a network installation. Therefore, the rule set can still function in a disconnected state (Figure 11-20 on page 365).



Figure 11-20 Lookup rule set populating the Vendor field from the fingerprint database

The IsStationIDSuffix action in the Invoice library examines the station ID of the users in Taskmaster and can conditionally run one of two OpenConnection actions that open a connection to a database. By default, these actions are set to the same value. However, the first function can be altered if you are working on a machine that is sometimes disconcerted to the fingerprint database.

The Execute SQL action in the Vendor field checks the fingerprint database and retrieves the fingerprint classification. In Accounts Payable Capture, this classification provides the vendor name, but in Flex, this classification provides the document type.

The VendorNumLookup rule set

The VendorNumLookup rule set must be customized when Accounts Payable Capture is deployed at a customer site. It ships with an Access database so that the demos can work.

In an installed system, IBM Taskmaster Accounts Payable Capture gets the vendor numbers directly from the business application system or from a recent copy of the vendor database of the business application system. The vendor number is not assigned to just a vendor name, but also to a vendor location, because many vendors have different addresses for the business application system to send checks to.

You might want the vendor number to be looked up based on the PO number. This strategy is also successful if the business application system can supply such data through a lookup.

Because this rule set must be customized, and a readily available version is identical in structure to the Lookup rule set, we do not address this rule set further in this chapter.

The Vendor NumberLookupClose rule set

The Vendor NumberLookupClose rule set falls into the same category as the VendorNumLookup rule set. The Vendor NumberLookupClose rule set ensures that you have closed off the connection to the vendor database in the production system.

The POLR rule set

The purpose of the POLR rule set is to try to furnish line item numbers to each of the recognized line items in the invoice. Similar to the previous two rule sets, this rule set must be customized to pull the line item data from the business application system of the customer.

POLR uses settings from the `settings.ini` file to do the work. In most cases, just altering these settings is sufficient to allow POLR to work at a customer site. Figure 11-21 lists the pertinent settings from the `settings.ini` file.

```
'If the Purchase Order Line item Reconciliation module is used, the following entries need to be modified
'for the customer supplied PO Database tables. The Test entries should point to the Datacap supplied tables.
*****
TestPODSN=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Datacap\apt\APTLook.mdb;Persist Security Info=False
TestPOLookup=Select (LineNo,QTY,ItemNo,Description) from POTable where PO = ?
*****
'The following two vendor references should point to the AP System vendor tables
*****
'For the POLookup value, you need to return the 5 items in the order listed.
PODSN=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Datacap\apt\APTLook.mdb;Persist Security Info=False
POLookup=Select LineNo,QTY,ItemNo,Description,Format(UnitPrice,'.00') from POTable where PO = ?
*****
FingerprintDatabase=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Datacap\apt\APTFingerprint.mdb;Persist Security Info=False

[POLR]
'A value of 1 in the following line item fields denotes that a match of the field is required for POLR to automatch. The
'AutoMatch feature requires an exact match of all fields marked with 1 to automatically match the PO lines.
ItemID=1
Qty=1
Price=1

'The price tolerance may be adjusted to allow for minor variations in the price based on rounding or truncation of decimals on the invoice.
'This tolerance is used for the automatching of line items only (when Price is set to 1).
PriceTolerance = .005

'POLR has the ability to write unused PO Lines to the Detail field as variables. A value of 1 enables this functionality. The SeparatorCharacter
'value is the delimiter. If Unspecified, then a pipe is used.
WriteUnusedPOLines=1
SeparatorCharacter=|
```

Figure 11-21 POLR settings stored in the `settings.ini` file

The first section in Figure 11-21 is the [Database] section. Similar to the Lookup rule set, POLR can also use a test database when it is being used in a disconnected state. The settings must point to the vendor business application

table that contains the PO information or the stored procedure that you use to retrieve the line items based on the current PO.

After the line items are successfully retrieved, POLR uses the ItemID, Qty, and Price keywords to determine which fields to use for automatch. In the previous example, lines automatch only if they have identical values for all three fields between the recognized line items and those retrieved from the business application system.

With manufacturing concerns, item prices are sometimes represented by a small fraction of the currency used. Therefore, a PriceTolerance is specified when automatching line items.

You might also want to output a list of any unused line items on a PO so that users can know that the invoice that they are paying does not close out a PO. For this reason, POLR also contains the capability to write the unused DCO lines to the DCO itself. Although the Accounts Payable Capture workflow does not immediately do anything with these line items, they are stored as variables on the Detail field of each Main_Page and can be exported if you want. Look at the detailed section in Figure 11-22 to see how unused lines from the PO are stored.

F	Details
	Text value :
	Char confi :
	TYPE : Details
	Position : 0,1097,2552,3295
	STATUS : 0
	Unmatched Invoice Lines : 3
	Unused PO Line 1 : 30 1100-NM Mileage Estimate 170.00
	Unused PO Line 2 : 40 1 Additonal Labor 400.00
	PreVerify Val :
	PreVerify Pos : 0,1097,2552,3295
F	Lineitem0
	Text value :
	Char confi :
	TYPE : Lineitem
	Position : 200,1102,2377,1147
	STATUS : 0
	PO Line Number : 20
	PO Qty : 1
	PO Item ID : 1000-NM
	PO Description : Base Labor
	PO Reconcilled By : auto
	PO Unit Price : 2100.00
	PreVerify Val :
	PreVerify Pos : 200,1102,2377,1147

Figure 11-22 How POLR stores data in the runtime DCO

The actual PO values are stored for each line at the line-item level in variables. The PO Linenumber is essential to export so that a three-way match can be easily and programmatically accomplished. However, some systems use other criteria, such as the description, when matching the lines for a three-way match. If so, consider exporting the line criteria that allows the three-way matching system to function the best. POLR allows the system and data entry operators to reconcile the lines to ensure that errors are at a minimum. The goal is also to supply additional data to the three-way matching system. This way the knowledge workers assigned to that process can concentrate on actual issues (items not received, incorrect counts, different prices) and do not have to match the lines against the PO.

Although POLR runs in the Batch Profile phase, it cannot do its job if the PO is initially misread. This task profile prematches line items as possible before verification. The data entry operator has an advanced POLR user interface to quickly match any line items that could not be matched in the background process.

The Validate rule set

The validate rule set is used to clean and format data. It is also used to apply business rules to ensure that data is the correct length and data type and is mathematically correct. Validation also checks to ensure that the data meets other business criteria before it can be successfully exported.

The Validate rule set runs in several task profiles. When running in the Batch Profiler Task Profile on a background machine before verification, this rule set formats and checks each field before a data entry operator views the document. The rule set then marks the problem fields. This way, the data entry operator knows that they need attention to pass the business rules.

This same rule set also runs when the data entry operator has viewed a document, potentially made corrections, and has submitted it as complete. Having the business rules in this single code base saves a lot of development and maintenance time with Taskmaster systems.

The Validation rule set in an Accounts Payable Capture workflow also employs enhanced error messaging (see 10.10, “Enhanced error messaging” on page 332). With this feature, you can write your own complete error messages that appear during the verify process if an operator submits a form that fails validation.

The length of this rule set makes it difficult to explain it rule by rule. Therefore, this section shows how different technologies are implemented within it.

Figure 11-23 shows a snippet of the Validate rule set that shows the page-level rule and a simple field showing Enhanced Error Messaging.

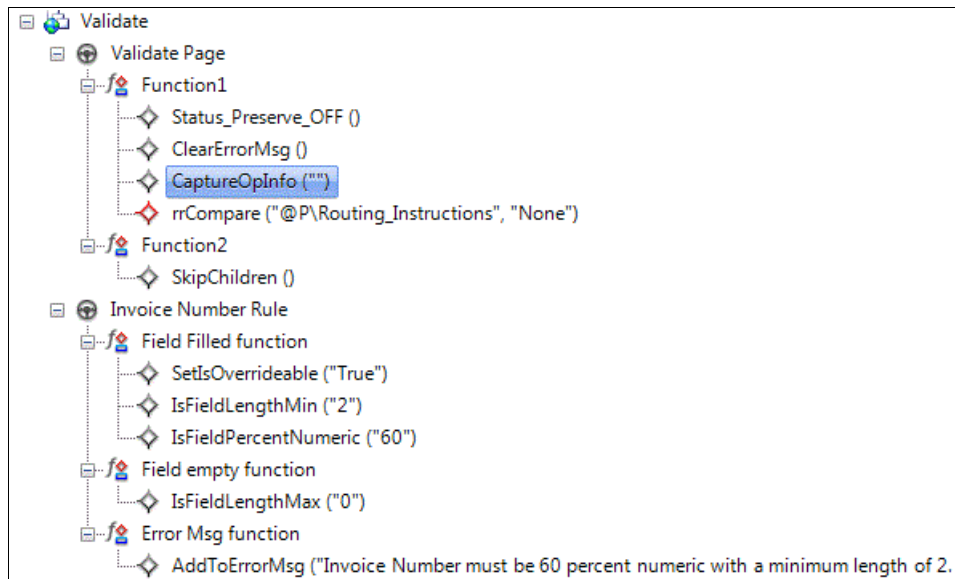


Figure 11-23 Snippet of the Validate rule set

In the Validate Page rule, first the Status_Preserve_Off() places the rules engine in a state where it sets the Page and every field on the page to a status of 0. At this point in the process, a status of 0 means that the fields have passed the validation rules and are not a problem.

The way Status_Preserve_Off() works is that the page and fields are set to a status of 0 and then the validate rules on each field run. If a field fails validation, it gets the problem status of 1. If it gets this status, the page also gets a status of 1.

Pages with a status of 1 are displayed to a data entry operator, while pages with a status of 0 are not displayed. Fields with a status of 1 are displayed in a pink color during verification. This way the data entry operator can quickly see by the color of the field that it failed the business rule associated with it. More information about status is provided later in this section.

The ClearErrorMsg() action is part of the Advanced Error Messaging employed by Accounts Payable Capture. A page-level error message variable that contains text from every field that failed validation is displayed when a data entry operator submits a form that fails any business rules. This action clears that variable before running the field validations.

The CaptureOpInfo() action writes the current operator and station information to the DCO for export at a later time, if desired.

The Routing_Instructions field is in the Accounts Payable Capture workflow. Normally this field is set to *None*, but an operator can identify documents that they cannot process for some reason by choosing a predefined routing instruction. The rr_compare() action at the end of the first page-level function (Function1) detects whether this field contains a value such as Delete, Rescan, or Review. If it contains anything other than None, the function returns *False*, and the second function (Function2) indicates to skip the validation rules on the fields.

For a sample field validation, see the Invoice Number Rule in Figure 11-23. Each rule can be set to *overridable* or *non-overridable*. Non-overridable fields must be corrected by the data entry operator. Otherwise, they are unable to proceed in verifying a batch.

For this field, a minimum length of 2, with data that is at least 60% numeric, is required according to the first function, the Field Filled function. If those conditions are met, the rule is finished processing, and the status on the field remains 0.

If this field fails the conditions of the first function, the Invoice Number Rule checks whether the field has a maximum length of 0, meaning that it is blank, according to the second function, the Field empty function. If this field passes that function, the rule is finished, and the status on the field returns 0.

If the first two functions fail, the Invoice Number Rule calls the Error Msg function, which calls the AddToErrorMsg() action. This action sets the enhanced error message by writing to the page-level variable. Therefore, because every function failed on the validation, the status of the field is set to 1, meaning it is displayed in pink to the data entry operator. Also the page status is set to 1, meaning that the page is displayed to the data entry operator.

Almost every field in the Accounts Payable Capture workflow contains similar validations that are altered for the business rules we want to apply to each individual field. Some of the fields have additional technology applied to them, in particular localization actions and mathematical validations. Figure 11-24 on page 371 shows the application of localization technology and calculations.

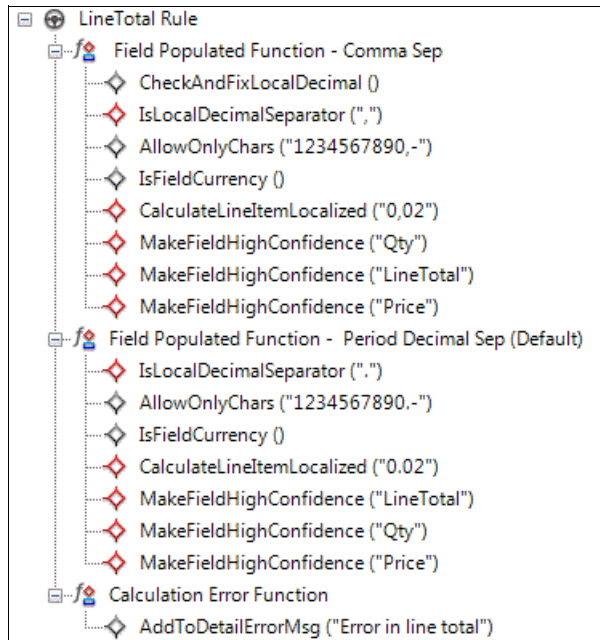


Figure 11-24 LineTotal Rule for the Validate rule set

The first action examines data and normalizes it to the decimal separator used on the local machine. At the time of writing this Redbooks publication, Accounts Payable Capture supports both commas and periods as decimal separators. This action examines the data in the field and changes the decimal separator to the same decimal separator set on the machine processing the rules.

Next, the LineTotal Rule checks this separator so that it can properly clean the data of any currency marks and any thousands of separators that are present. Depending on the local decimal separator, different characters are allowed.

The LineTotal Rule then checks to ensure that the field is a currency field. Then it mathematically calculates the line item by multiplying the Qty by the Price and ensuring that it equals the line total, within the tolerance specified as a parameter. Having a tolerance in many applications is important because the document might limit the number of decimal places that are displayed to fewer than what is required for an exact equivalence match.

An interesting technique that you can sometimes employ follows the calculation. If the calculation is correct, the field values must also be correct, even if they contain low-confidence characters. In this application, the fields involved in the calculation are marked as high confidence because we know that they are correct based on the calculation.

The Routing rule set

In general, the Routing rule set is used to check the batch for low confidence characters and problem fields to determine whether the batch must go to a data entry operator. However, with the volume of data in an invoice batch, it is improbable that the data entry step can be skipped completely. Therefore, the Routing rule set is used to do final preparation of the batch before a data entry operator sees it. Figure 11-25 shows the Accounts Payable Capture implementation of the Routing rule set.

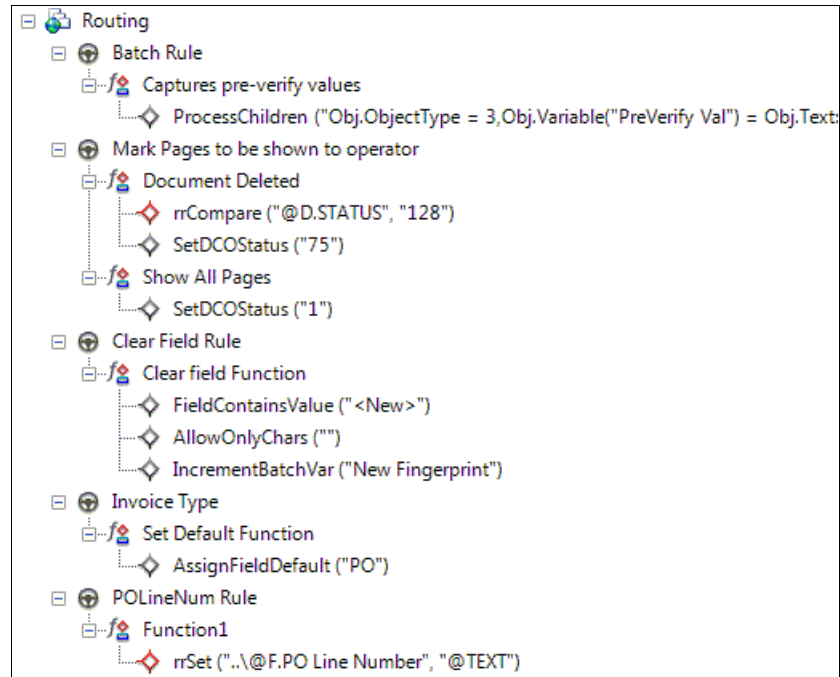


Figure 11-25 Routing rule set

The Batch rule uses a ProcessChildren action to capture the preverify value and position of each field and to store it as a field-level variable. The preverify position variable is used by Intellocate to determine if a field was zoned by the data entry operator. You can use the preverify value to capture statistics on the number of fields that a data entry operator changed, but it is not available immediately.

The page-level rule called *Mark Pages* can be shown to an operator. The first function checks whether the document has been deleted. Deletion of the document can only happen if a page in the document is not recognized. If the page is not recognized, the page is marked with a status of *deleted*.

All other Main_Pages are marked with a status of 1, meaning that the data entry operator sees every Main_Page in the batch. Most people prefer to see every page as a visual check that everything is working before exporting the data and the image.

If a document is new to the system, the Vendor field is populated with the value <New> during the Lookup rule set. The Clear field rule erases this text from the field and leaves it blank. It also increments a batch variable so that you can have a record of how many new fingerprints were in this batch.

The Invoice Type rule defaults the Invoice_Type field to a value of PO. The POLineItemRule copies the POLR variable for the PO Number to the text property of the appropriate field.

11.5.3 The Verification process

Several rule sets run under task profiles during the verification process. In verification, task profiles can be set up to run automatically or when a data entry operator clicks a button. This section highlights the following rule sets that can run under task profiles called by the verification process:

- ▶ The DynamicDetails rule set
- ▶ The CheckForSticky rule set
- ▶ The AutoCalc rule set

The DynamicDetails rule set

The DynamicDetails rule set is a way to find, at verify time, all of the line item fields in a document. You click the line-item subfields of the first detail line on the invoice and click the **Find Details** button. This rule set sets up the Lineitem and Detail zones automatically.

The DynamicDetails rule set runs under the Find Details Task Profile and is called by the **FindDetails** button on the **Verify** tab. Figure 11-26 shows the FindDetails Task Profile.

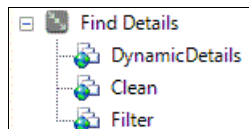


Figure 11-26 Find Details Task Profile

The DynamicDetails rule set (Figure 11-27) is identical to the rules that are associated with the Detail level and Lineitem rules in the Locate rule set. However, special actions are needed to read from a CCO that was loaded dynamically at verify time. The actions function identically to their non-dynamic counterparts.

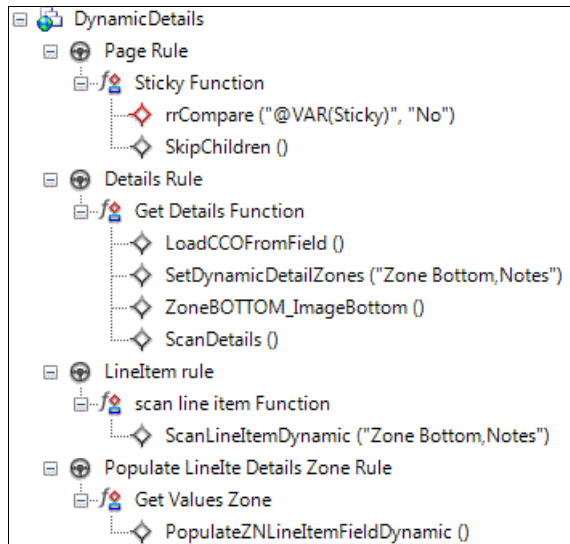


Figure 11-27 DynamicDetails rule set

The Clean and Filter rule sets called in the Task Profile are the same rule sets that are called in the Batch Profiler Task Profile.

The CheckForSticky rule set

The CheckForSticky rule set runs in the StickyFingerprints Task profile as shown in Figure 11-28. As explained previously in this chapter, you must be familiar with the other rule sets that are called in this task profile.

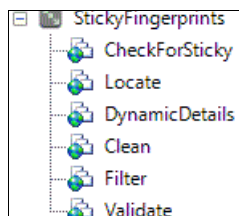


Figure 11-28 The StickyFingerprint Task Profile

The StickyFingerprint Task Profile runs automatically from a thick client when it detects that a new fine-grained is being processed and another of the same new

fingerprint was processed previously in the same batch. For example, a vendor sends in two invoices with a new format. The first one creates a fingerprint, and the second one matches that same new fingerprint. At verify time, the operator zones the first invoice. When the second invoice is displayed, the StickyFingerprint rule set copies and adjusts the zones from the first invoice to the second invoice. Then it automatically populates with data.

If you are reading this chapter from the beginning, you are familiar with all of the rule sets called by StickyFingerprint, except for the CheckForSticky rule set (Figure 11-29).

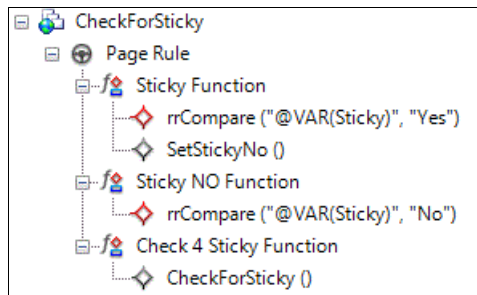


Figure 11-29 CheckForSticky rule set

When you first enter this rule set, the Sticky variable must be blank. The first two functions fail, and the CheckForSticky action runs. This action checks whether previous documents were in the batch that can be used to zone the current document. If there are such documents, this action adjusts and copies the zones to the new document. It also sets the Sticky variable to *Yes* or *No* depending on what it detected when it analyzed the batch. If the variable is *yes*, the other rule sets in the Task Profile run, and the data is populated. If the variable is set to *no*, the other rule sets are coded to do nothing.

The first two functions of this rule are in place if Sticky is run for a second time for some reason. If the variable coming into this rule set is already set to *Yes* or *No* instead of blank, the CheckForSticky rule does not run.

The AutoCalc rule set

The AutoCalc rule set (Figure 11-30) runs in the CalculateBlank Task Profile. It is called when the verify operator clicks a button. Because the Qty, Price, and LineTotal fields are mathematically related, this rule set enters a single missing value if it is detected on a line item.

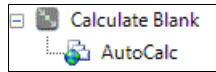


Figure 11-30 The CalculateBlank Task Profile showing the AutoCalc rule set

The CalculateBank Task Profile calls a single rule set to do the analysis and automatic correction of a blank Qty, Price, or LineTotal field on a Lineitem as shown in Figure 11-31.

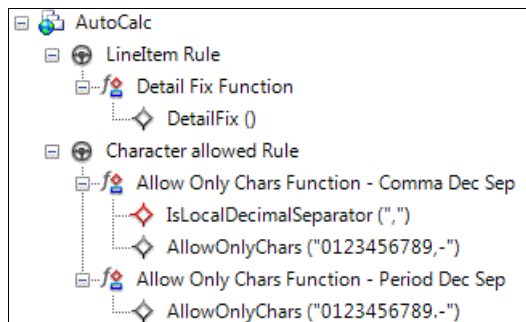


Figure 11-31 The AutoCalc rule set

The Character allowed Rule is applied to each field to ensure that the values in them are normalized to the proper localization. Then, the DetailFix Rule is applied to calculate a single missing value on each line item.

11.5.4 The Export Task Profile

The Export Task Profile consists of five rule sets that are ready for immediate use. However, it does not contain rule sets to export the data to your imaging system or to your business application system. Because these rule sets are highly variable, the demo writes the data to an XML file. You must add additional rule sets to this task profile for a production installation.

Not all of the rule sets in this Task Profile are used to export data. Many of the rules are for preparing the data for an export. Other rules are for handling the disposition of problem documents, such as those documents that must be

reviewed or rescanned. The Task Profile also includes the Intellocate rule and a rule to capture fingerprint statistics.

The Export Task Profile includes the following rule sets as shown in Figure 11-32:

- ▶ The SetStatuses rule set
- ▶ The PreExport rule set
- ▶ The Export rule set
- ▶ The ExportClose rule set
- ▶ The RoutingNotification rule set

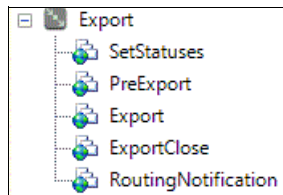


Figure 11-32 Export Task Profile

The SetStatuses rule set

Two systems are available for marking documents for deletion, rescan, and review. The thick client Verify task does this task with user keystrokes and sets statuses on the documents and the pages for you. The thin client verify panels rely on a drop-down list in the field `Routing_Instructions` to mark documents for the same issues. The SetStatuses rule set (Figure 11-33 on page 378) consolidates the two methods. This way, at export time, you only have to check the statuses or the `Routing_Instruction` field to determine if you want to export them or send a notification to someone.

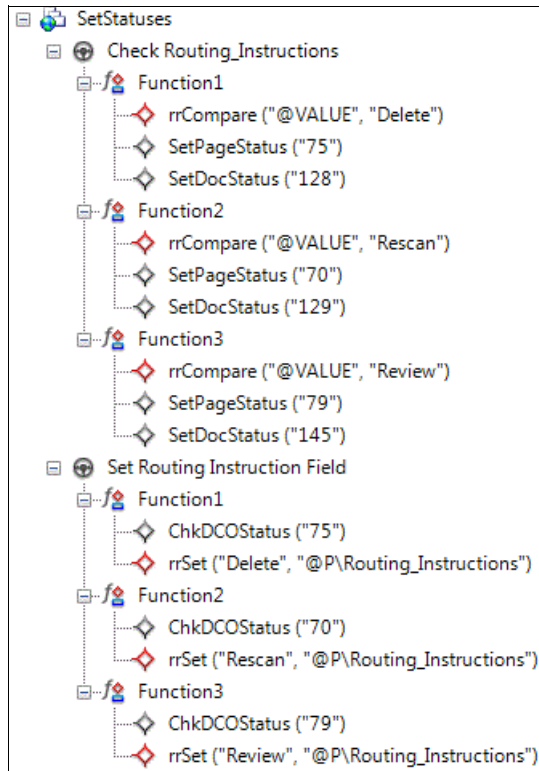


Figure 11-33 The SetStatuses rule set

The first rule, Check Routing_Instructions, runs on the Routing_Instructions field. It looks at any value other than the default value of None and sets the statuses on the page and document accordingly.

The second rule runs on the page level and checks the current page status that a thick verify client might have set. If the current page status is set, the rule sets the Routing_Instructions field to the appropriate value.

Because of this rule set, you only need to check one of the methods for marking documents for special handling with the rest of the Export rule set.

The PreExport rule set

The PreExport rule set is a catch-all for everything that needs to be done before exporting the documents and images. It is also where you can find Intellocate, one of the most important features of DNA applications.

Figure 11-34 shows the non-Intellocate rules that are associated with the PreExport rule set.

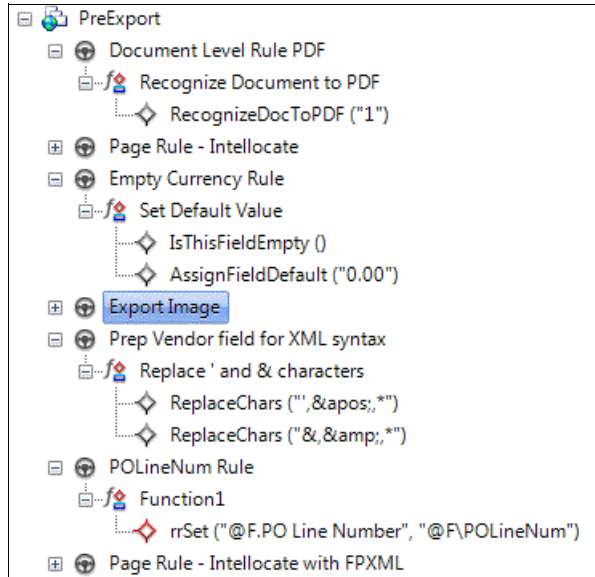


Figure 11-34 Non-Intellocate rules in the PreExport rule set

The document-level rule uses the Scansoft recognition engine to make a text searchable PDF at the document level. This PDF is displayed in the batch directory and is named with the DocumentID property and a .pdf extension.

The empty currency rule attaches to each field that contains currency and defaults the value to 0.00 if the field is blank. This rule might need to be changed to 0,00 if a comma is used as a decimal value in the system to which you are exporting.

The Prep Vendor Field for XML Syntax checks the vendor field for an apostrophe (') or an ampersand (&) and replaces those characters with the XML equivalents to those characters. The PO Lineitem rule ensures that the PO LineNumber field is populated with the data from a POLR lookup.

The Intellocate rule set makes up the bulk of the processing in the PreExport Task profile. Figure 11-35 shows the first part of the rule set, Page Rule - Intellocate rule. The rule runs on the Main_Page object.

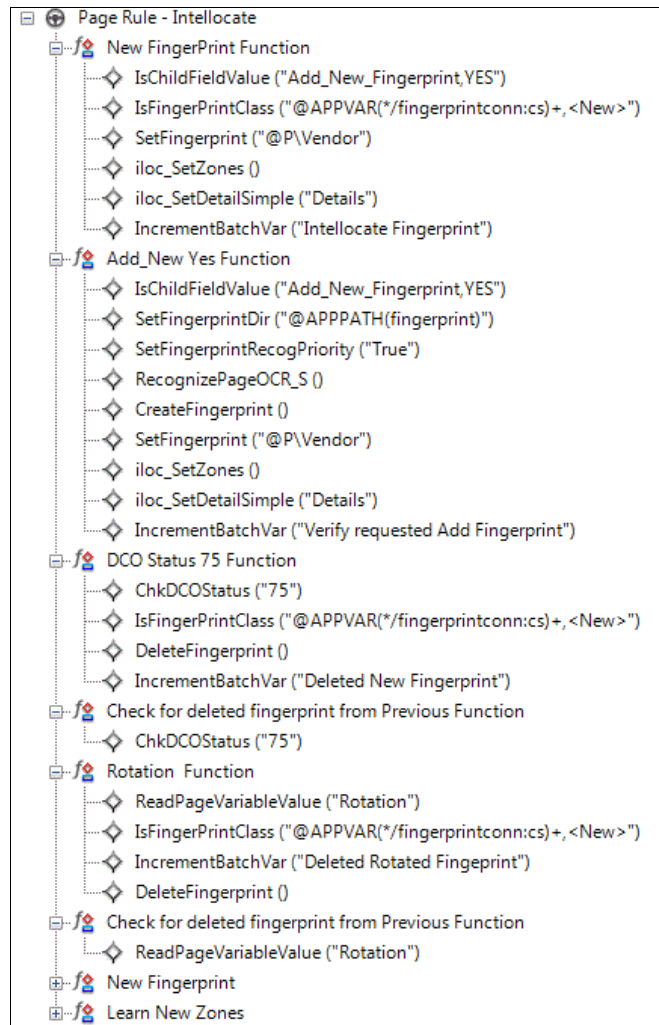


Figure 11-35 First part of the Intellocate rule set

The first two functions control what happens if a data entry operator defines to the system the need for a new fingerprint based on the current image. Normally this action is only done if a fingerprint mismatched. However, sometimes data entry operators mistakenly choose to add a fingerprint when the fingerprint is already in the <New> classification, which is created anyway.

For the first function to complete successfully, the operator must choose **NewFingerprint** when it is presented to them. The fingerprint is in <New> to correct a misunderstanding that some operators have. If the fingerprint is in <New>, Intellocate saves the zones for the fingerprint. We do not want create an additional new fingerprint. If this happens, the **SetFingerprint** action classifies the fingerprint (moves it out of the <New> classification) into a classification that matches the Vendor name. The **iloc_SetZones** sets the header fields, and the **iloc_SetDetailsSimple** sets the zones of the detail fields.

Important: These actions write the zones to the Setup DCO. If you want to use FPXML, unhook this rule from the page and replace it with the unattached rule in this rule set for FPXML.

In its essence, Intellocate is done with three actions: **SetFingerprint**, which moves the fingerprint out of <New> and classifies it, and the two **iloc** actions that save the zones. The rest of these actions ensure that these actions need to be done.

The second function runs if the data entry operator clicked the **NewFingerprint** button correctly this time because the document matched an existing fingerprint erroneously. When this happens, we want to dynamically create a fingerprint in the library from the existing image, run Intellocate on it to classify it, and save the zones.

Because the page might have a multi-CCO (MCCO), we re-recognize the page to ensure that it creates a single page CCO for the first page. We also indicate where to store the new fingerprint that it creates. We use **CreateFingerprint** to create the fingerprint before we use Intellocate.

The DCO Status 75 function handles deleted documents. If an operator indicates that a document does not belong in the system, we want to delete any fingerprint that it created. This function can return a value of *false* if the fingerprint was somehow already deleted. Therefore, the trailing function checks this value and returns *true* so that the rule does not continue.

The same process is done for rotated documents. You do not want to store upside down fingerprints. Therefore, this function deletes them.

If a fingerprint remains in <New> after these functions have checked for special processing, the New Fingerprint function runs (Figure 11-36).

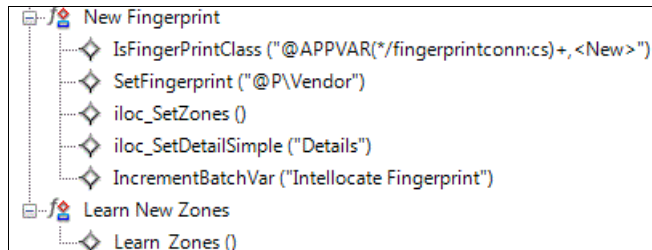


Figure 11-36 Remaining part of the Intellocate rule

As you can see, this process takes any fingerprint that is still in <New>, automatically classifies it, and saves the zones.

If a fingerprint is not in <New>, the LearnZones action examines the PreVerify Position variable that was created in the Routing rule set and the current position of the field in the runtime DCO. If the PreVerify Position was 0,0,0,0 and the operator provided a zone for the data, the position of that field is added to the other fingerprint positions.

The Export rule set

When you demo Accounts Payable Capture, you might not have access to a business application system or an imaging system. Therefore, for demo purposes, we write out a standard text file with XML tags. Because this rule set is not commonly used for production, it is not explained here. This file writes the text searchable PDF documents and data in XML format in the APT/Export directory.

The ExportClose rule set

The ExportClose rule set writes the XML tags to complete the XML file and closes the file. This rule set is not used in production.

However, you want to put the SetFPStats() action in the ExportClose rule set in your custom export. This action counts the number of times and the most recent time that a particular fingerprint was used. It helps you to manage the fingerprint library to remove fingerprints that are no longer being used by any of the vendors (Figure 11-37 on page 383).

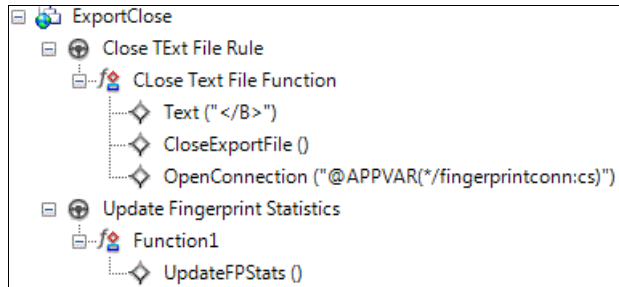


Figure 11-37 UpdateFPStats action that must be in your export

The RoutingNotification rule set

The RoutingNotification rule set is also altered for production. This rule set must be in place if you want to notify someone that the batch had documents in it that could not be processed (Deleted, Rescan, or Review).

Figure 11-38 shows the RoutingNotification rule set.

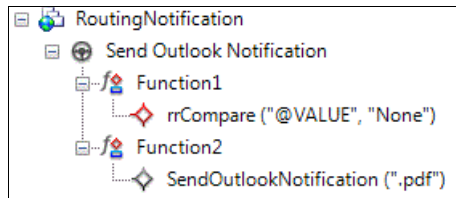


Figure 11-38 RoutingNotification rule set

The first function checks the Routing_Instructions page. If it is not set to *None*, it fails. The failed document is then processed by SendOutLookNotification that sends an email to a person specified in the settings.ini file and attached the multipaged PDF file.

The reason that this action is not used in production is that it requires Microsoft Outlook to be set up on the background machines. Also the version of Microsoft Outlook must support sending email messages programmatically, but not versions do. Therefore, replace this action with another email action to send the documents. Alternatively, develop another method of informing someone of these problem documents that is an entry in a database.



Part 4

Application and performance

This part addresses high availability, scalability, performance, and backup options for the document imaging solution. In addition, it briefly explains Taskmaster software installation, application migration, and reuse.

This part includes the following chapters:

- ▶ Chapter 12, “System scalability, availability, backup, and recovery” on page 387
- ▶ Chapter 13, “Installation, migration, and application reuse” on page 453



System scalability, availability, backup, and recovery

This chapter describes high availability, scalability, performance, backup, and recovery options and strategies for IBM Production Imaging Edition, specifically IBM Datacap Taskmaster Capture (Taskmaster).

This chapter includes the following sections:

- ▶ System scalability, performance, and availability
- ▶ Rulerunner
- ▶ Backup and restore
- ▶ Configuring Rulerunner
- ▶ Adding additional Taskmaster Servers

12.1 System scalability, performance, and availability

Building an enterprise solution across multiple locations, with large user numbers and a high volume of document throughput, requires a system that can scale horizontally and vertically. This system must also be able to withstand node failure.

Production Imaging Edition is made up of two key components: Taskmaster to scan and capture images and IBM FileNet P8, which is the central repository to store these images. Both Taskmaster and FileNet P8 have several options to design a scalable and available system. This chapter concentrates on scaling the Taskmaster component of Production Imaging Edition.

For information about scaling the FileNet P8 component, see the IBM Redbooks publication *IBM FileNet P8 Platform and Architecture*, SG24-7667, which explains the approaches and methods of scaling, performance tuning, and availability.

12.1.1 Typical Datacap installation

A typical Taskmaster implementation for a small scale project consists of the following servers:

- ▶ The Taskmaster Server, which is the central service that provides user authentication, workflow and queuing
- ▶ A Rulerunner Server that runs tasks that do not require human interaction such as Optical Character Recognition (OCR), Intelligent Character Recognition (ICR), and export to repository.

Figure 12-1 shows a simple Taskmaster architecture.

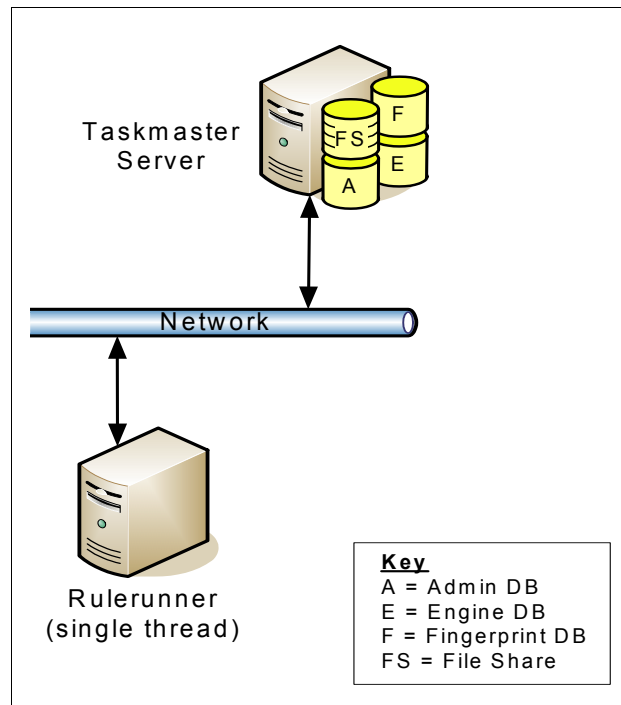


Figure 12-1 Simple Taskmaster architecture

Although it is possible to run all Taskmaster components on the same physical machine, perform this task only for the smallest production capture systems and for development configurations. Performance is limited because Rulerunner tasks are typically processor-intensive and can interfere with Taskmaster Server response times.

12.1.2 Scaling Rulerunner vertically (scale up)

Taskmaster can be scaled vertically, which you can achieve in the following ways to handle the volume of documents that you want to process:

- ▶ Increase the hardware specification of the Rulerunner server so that processes can run faster. To realize this goal, you can perform such tasks as upgrading processors, increasing memory, and obtaining a faster network connection.
- ▶ Increase the number of available Rulerunner threads to allow multiple tasks to process concurrently and use multicore processors more efficiently. For more information about this process, see 12.2, "Rulerunner" on page 402.

Multithread licensing: At the time of publication, an additional license, called the *Rulerunner Enterprise license*, is required to use multithreading in Rulerunner. Contact your IBM marketing representative for more details.

There is a limit to which you can scale the hardware configuration of a single server. The limit can be a physical limit. For example, you cannot get faster components. You can also reach a point at which the benefits of upgrading hardware are less, such as in terms of cost versus performance, than the purchasing of a new separate machine.

Rulerunner is a 32-bit application and, therefore, is restricted to 2–4 GB of addressable memory. However, Rulerunner creates separate processes for OCR and other intensive tasks, which allows for usage of additional memory if the operating system is 64 bit.

Vertical scalability relies on adding processing power to a physical machine. Configuring a system with just one Rulerunner server implies a single point of failure for background processes.

12.1.3 Scaling Rulerunner horizontally (scale out)

Taskmaster can be scaled horizontally by increasing the number of Rulerunner servers used to handle a high volume of documents.

In an installation of this type, you have a centralized Taskmaster Server and a defined number of Rulerunner servers for unattended processing of ingested documents. Depending on your license, you can configure these servers to run a single thread or multiple threads. For more information, see 12.2, “Rulerunner” on page 402.

Figure 12-2 shows a typical horizontal configuration, with a centralized Taskmaster Server and two single-threaded Rulerunner servers that share the load of document processing.

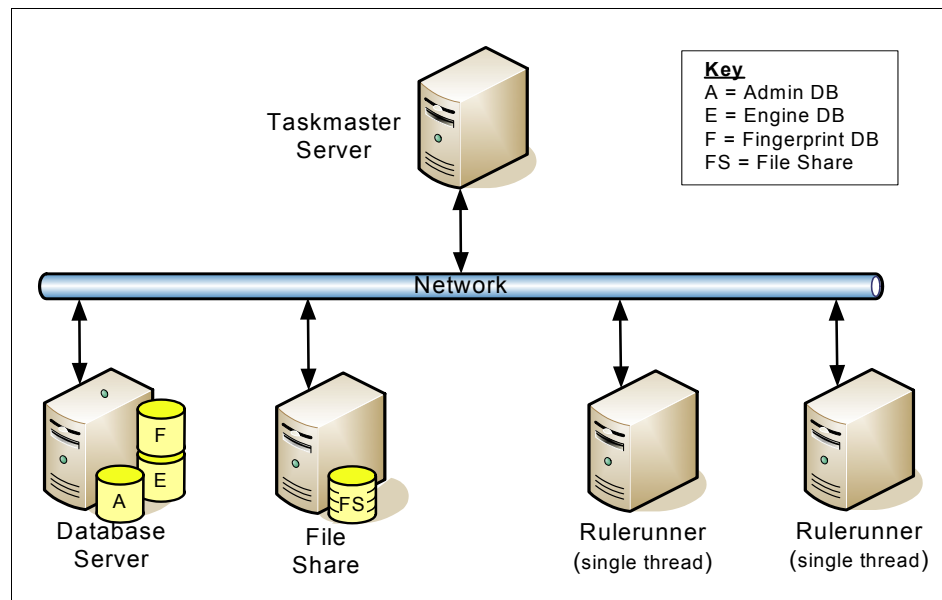


Figure 12-2 Typical horizontal configuration

Each Rulerunner server uses the first-in first-out (FIFO) algorithm to poll the Taskmaster Server for the oldest batch of documents pending for a set of defined background tasks. Then the Rulerunner server processes each batch in turn.

Rulerunner can be configured to poll for specific tasks in given projects. To scale effectively, tasks must be shared intelligently across the available Rulerunner servers. For most purposes, as a best practice, configure all Rulerunner servers to run all background tasks to ensure that any work is processed by the first available Rulerunner as soon as possible. If a Rulerunner server fails, the remaining Rulerunner servers continue to process all tasks. Alternatively, you might want to assign specific subsets of work to a subset of Rulerunner servers. This method helps primarily to control the amount of processing power assigned to specific parts of the workflow. For example, it ensures that priority is given to certain tasks, even if older work is available for other tasks.

The scale-out approach has one disadvantage. That is, as you increase the number of single-threaded Rulerunner servers, you increase the number of physical or virtual machines that require management and administration.

A trade-off between performance, cost, and resiliency must be determined on a case-by-case basis.

For more information about the process of assigning tasks to Rulerunner servers, see 12.3.2, “Installing a single Rulerunner server” on page 412.

12.1.4 Scaling Rulerunner horizontally and vertically

Taskmaster can be scaled horizontally and vertically to achieve the advantages of both methods:

1. Scale up by upgrading the hardware specification of the Rulerunner servers to maximize single-server throughput to maximized hardware usage on the server.
2. Scale out by adding additional Rulerunner servers, with each server running a single threaded Rulerunner.
3. Increase the number of processing threads that are available to each Rulerunner by upgrading to the Rulerunner Enterprise license.

Figure 12-3 illustrates a horizontally and vertically scaled Taskmaster implementation. It details the usage of single- and multi-threaded Rulerunner servers. It also shows usage of a separate database server, file share, and dedicated fingerprint service. More information about these components is provided later in this chapter.

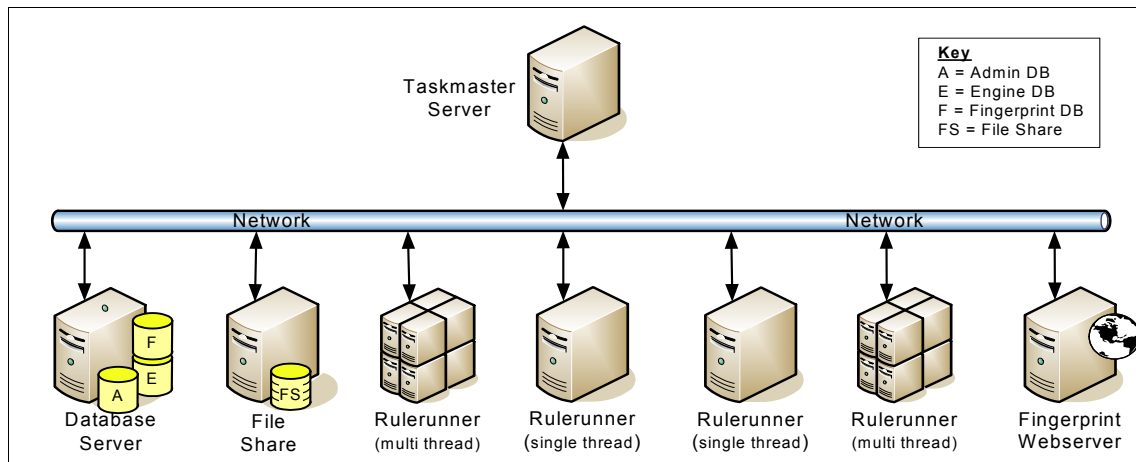


Figure 12-3 Horizontally and vertically scaled Taskmaster configuration

With this approach, you have the advantages is that you remove single points of failure for processing tasks and maximize the performance of available hardware.

For a walkthrough of a Rulerunner configuration, see 12.3, “Configuring Rulerunner” on page 409.

12.1.5 Taskmaster Server scaling and redundancy

As explained in the previous sections, you can scale Rulerunner horizontally and vertically. You can achieve this scaling by increasing the number of Rulerunner servers, increasing the hardware specification, and increasing the number of processing threads by the addition of a Rulerunner Enterprise license. This approach is suitable for task processing, such as OCR or validation. However, all the batches that are being processed are managed by the Taskmaster Server.

The Taskmaster Server is the central Windows service. It provides user authentication, workflow, and queuing (and file services for Taskmaster Web). Without this core component, no tasks can be managed and updated in the system.

Typically the load on a Taskmaster Server is low. Processor and I/O-intensive processes, such as OCR and export, are run on the Rulerunner servers. The main function of Taskmaster Server in this scenario is to access the Taskmaster queuing database (the Engine database) to select batches for processing and update statistics. Roughly six database transactions are performed for each batch. As a result, moderate loads can be easily supported with a single Taskmaster Server. The load must be quite high to require use of an additional server.

This rule comes with exceptions. For example, when using Taskmaster Web to permit the use of web clients, an increased load is placed on the server to transfer batch files to and from the web server. For more information, see 12.1.7, “Taskmaster Web scaling and redundancy” on page 397.

Also, using the RV2 Report Viewer tool increases the load on Taskmaster Server to transfer data from the database to the web server and RV2. If large data sets are transferred, this added load can be significant and might necessitate scale out of Taskmaster Servers.

If the hardware limitation of the Taskmaster Server is reached, in the first instance, the server hardware can be upgraded to meet the demand. If the server hardware does not meet the requirements, additional Taskmaster Servers can be installed. Rulerunner servers and Taskmaster Web can be configured to access the additional Taskmaster Server.

Active-active configuration

Figure 12-4 shows two or more Taskmaster Servers that are configured as companion servers.

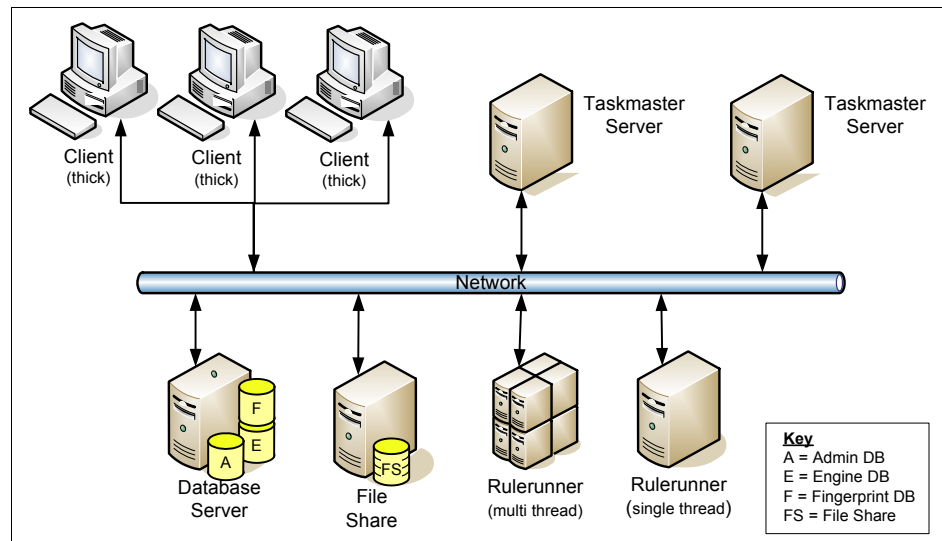


Figure 12-4 Active-active Taskmaster Server

The Rulerunner servers (single-threaded or multithreaded) can poll either or both Taskmaster Servers separately, looking for tasks to process.

As of this writing (for Taskmaster v8.0.1), all batch creation tasks must be configured or operated so that they can access a single Taskmaster Server. Conflicts between multiple Taskmaster Servers creating new batches can result in skipped batch IDs and delays in batch creation on some servers.

If one or more Taskmaster Servers fail, the clients connect to the remaining Taskmaster Servers. However, each Rulerunner server must be configured first to achieve this connection. For more information, see 12.4.1, “Adding a failover Taskmaster Server” on page 434, and 12.4.2, “Load sharing between Taskmaster Servers” on page 439.

The Taskmaster Servers use the same database servers and Universal Naming Convention (UNC) path so that the same project can be shared across multiple Taskmaster Servers.

Active-passive configuration

In an active-passive configuration, a single Taskmaster Server is configured as the primary server. A second Taskmaster Server is configured on a secondary server with the same parameters as the primary server, meaning the same IP address and server name.

If the primary Taskmaster Server fails, the secondary server is started manually. Because its configuration is identical to the failed server, it begins to process in the same manner as the primary server. All clients and Rulerunner servers connect as though they are the primary server.

Automatic starting option for the secondary server: The secondary server can also be started automatically by using Microsoft Cluster Services. Although this method has not been tested by IBM, customers are using this configuration successfully.

Batch abandonment and rollback

In the active-active and active-passive configurations, if a Taskmaster Server fails, any tasks that are processed by clients connected to that server are abandoned in the running state. The secondary server resets such batches before reprocessing.

Batch reset, also known as *rollback*, can be performed manually or through a scheduled NENU utility. In certain cases, you must restore the former state of the batch for proper reprocessing. This topic is beyond the scope of this book and is not covered here.

12.1.6 Scaling both Taskmaster Server and Rulerunner

Figure 12-5 shows a possible configuration of Taskmaster Servers and Rulerunner servers.

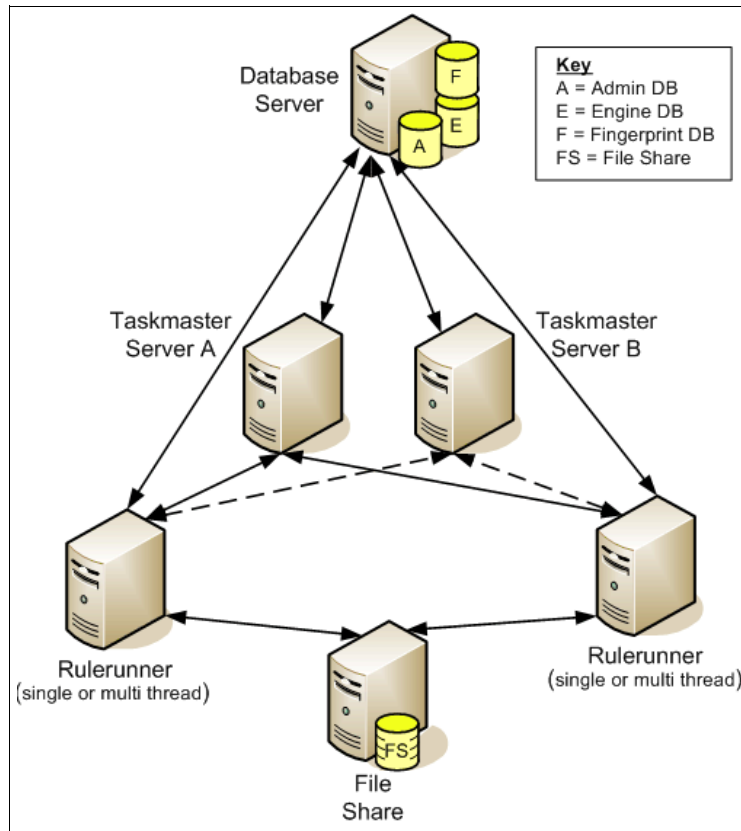


Figure 12-5 Taskmaster and Rulerunner configuration

The two Taskmaster Servers are connected to a single database server. With this connection, both servers can serve out tasks and batches from the same application, and each server updates the batch status to a common database. If one server fails, the other server still has access to the database and batch status.

Rulerunner (single-threaded or multithreaded) has been configured to poll one specific Taskmaster Server considerably more than the other (primary higher priority server). However, it occasionally polls the secondary Taskmaster Server (secondary lower priority server). The fact that Rulerunner knows about both Taskmaster Servers means that, if one fails (the primary high priority server in

this example), it starts using its secondary lower priority Taskmaster Server. It drops its priority and polls the secondary Taskmaster Server for all allocated tasks it has configured to process.

12.1.7 Taskmaster Web scaling and redundancy

Taskmaster Web relies on the use of Internet Information Services (IIS) for Windows Server to serve out a web-based user interface for administration, scanning, and data verification tasks. Because the IIS might reside outside of a firewall, for security purposes, all file and database requests are routed through the Taskmaster Server. This routing ensures that all such requests are run securely within the firewall. It also places a greater load on Taskmaster Server than a thick client.

Taskmaster Web response times depend on Taskmaster Server performance. When many users run tasks with large amounts of I/O, such as image upload and database lookup, implementations, including Taskmaster Web, must pay attention to the specification and configuration of Taskmaster machines to ensure adequate performance.

Rules, such as validation and document integrity, that are triggered during Verify tasks are also run on the IIS under Taskmaster Web. Heavy use of validation or other rules also increases the load on the IIS.

If the number of concurrent users who are using Taskmaster Web exceeds the capability of IIS, add another separate IIS that points to the Taskmaster Servers.

If the throughput generated by Taskmaster Web is too great for a single Taskmaster Server, you can add additional, separate Taskmaster Servers in an active-active configuration. For more information, see 12.1.5, “Taskmaster Server scaling and redundancy” on page 393.

Typical capacity range from 50 to 100 users for each instance of IIS.

Figure 12-6 shows a Taskmaster Web configuration that uses multiple Taskmaster Web Servers and multiple Taskmaster Servers.

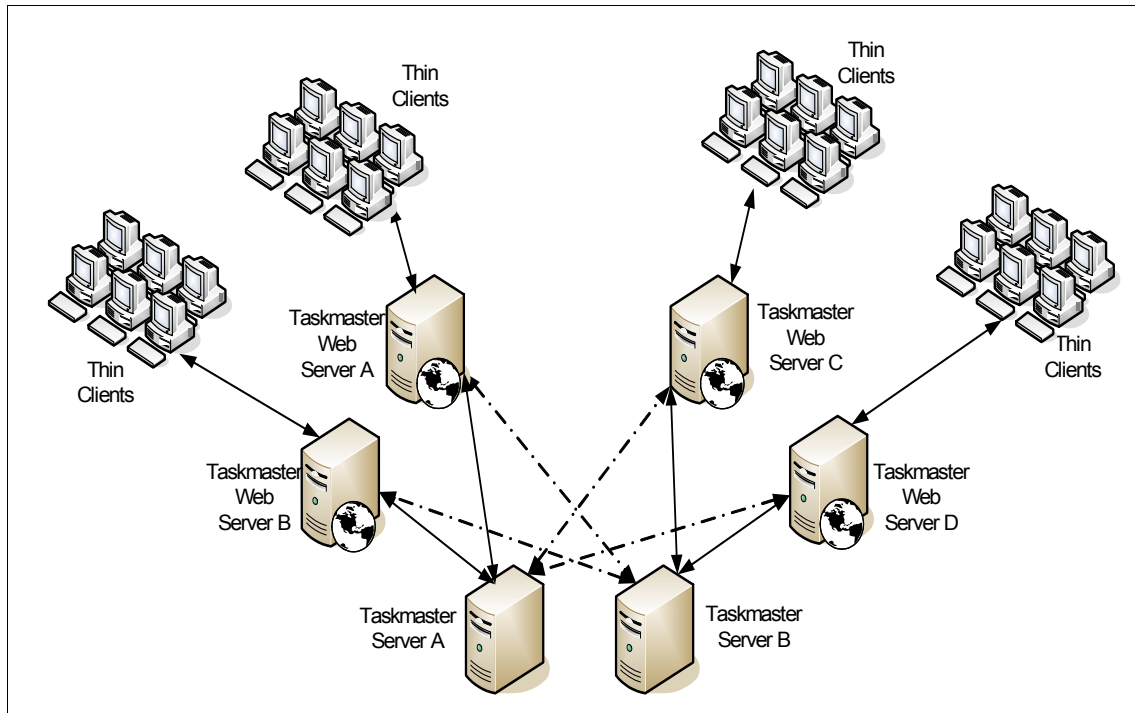


Figure 12-6 Multiple Taskmaster Web Servers with multiple Taskmaster Servers

The load of web-based thin clients is balanced among the Taskmaster Web Servers to which they connect. Each Taskmaster Web Server can be connected to a primary and secondary Taskmaster Server. This connection allows load balancing between the Taskmaster Servers and adds an element of failover if a Taskmaster Server fails.

12.1.8 Scaling and redundancy for thick clients

Thick clients connect to the Taskmaster Server to authenticate, start batches, verify pages, perform administration, or invoke several other tasks that are not automated.

A thick client connects to the primary Taskmaster Server as defined in the .app file of the application to which it is trying to connect. If this Taskmaster Server fails, the client connection drops.

To reinitiate a connection, the client must be restarted. Upon restarting the client again, it first tries to connect to the primary Taskmaster Server. If this server is still down, the client then attempts to connect to a secondary Taskmaster Server (if configured).

It is also possible to define which Taskmaster Server a client connects to as a primary server. This approach can help to scale out with additional thick clients beyond the capacity of a single Taskmaster Server.

Figure 12-7 shows the configuration of Taskmaster thick clients connecting to a primary Taskmaster Server (Taskmaster Server A). It also shows the redundancy to connect to a secondary Taskmaster Server (Taskmaster Server B).

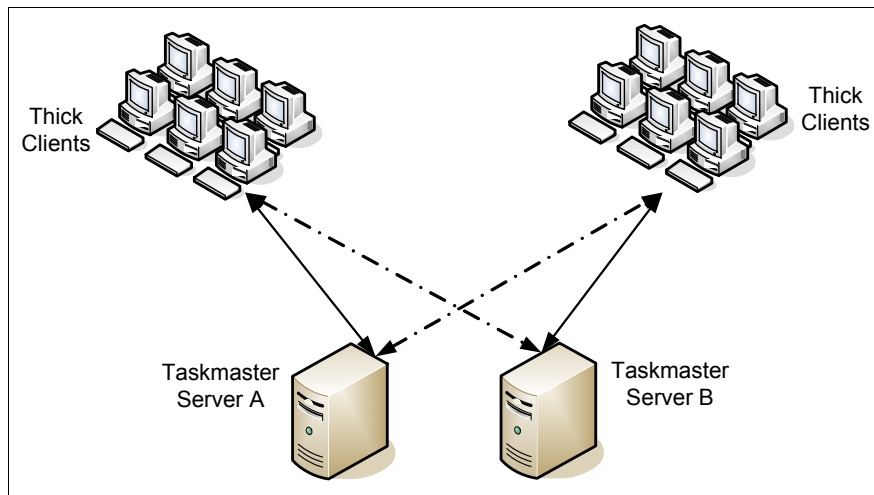


Figure 12-7 Taskmaster thick clients connecting to multiple Taskmaster Servers

12.1.9 Load balancing of tasks

Load balancing of unattended or background tasks is done in a simple way. Each Rulerunner thread polls the Taskmaster Server that is requesting a list of batches that are pending for any one of the tasks configured. If a batch is available, the first thread to request it gets the work.

Taskmaster processes batches according to their priority level (1–10, with 1 being the highest priority) and then by age (FIFO). Batch priority can be assigned manually or by rules at any point in the workflow.

All Rulerunner servers poll at regularly defined intervals.

Rulerunner has a facility to place a polling priority on each task. The task polling priority is separate from the priority assigned to the batch. The polling priority for a task defines the number of times that a thread polls for the task. For more information, see “Mixed queuing mode” on page 406.

12.1.10 Scaling databases

Taskmaster comes with Microsoft Access as its standard shipped database format. The two main databases are the Taskmaster Engine database and the Taskmaster Administrator database. The Taskmaster Administrator database holds the user credentials for users of the system. The Taskmaster Engine database maintains the status of each batch in the system.

For production implementations, use enterprise-level databases that are currently supported by Datacap. Reserve use of the Microsoft Access database for demonstration, low workloads, or simple testing environments.

To provide optimum performance and scalability, databases must be on a separate dedicated server primarily used for this purpose. For small production systems, the database can share a server with Taskmaster Server or Rulerunner.

Scaling of enterprise-level databases is documented. Because it is beyond the scope of this IBM Redbooks publication, the topic is not addressed in this book.

12.1.11 Network share drive

When Taskmaster captures and processes documents, it stores them in a shared file system. This file system must be read/write accessible across the network by all Taskmaster and Rulerunner servers.

Because batches contain multiple image files, the network bandwidth required for moving these files can get high. Therefore, sufficient bandwidth must be available between the client and the network share drive. The file system must also be mounted on a system that is capable of fast input and output.

A common configuration is to have a Taskmaster Server fail over with a storage area network (SAN) drive that is connected to both.

Small scale implementations suffice on a desktop or on a small server system. For larger scale systems, with high throughput requirements, use a fully dedicated high-speed disk subsystem. Batches of documents can be archived to back up media after they complete the workflow. Archiving can be achieved by using the NENU utility.

12.1.12 Scaling across geographies

Scanning of files and storing them in a remote network shared drive can present performance issues if the network is relatively slow. Therefore, where possible, store the files on a local network share. Figure 12-8 shows a possible distributed, thick client implementation of IBM Production Imaging Edition.

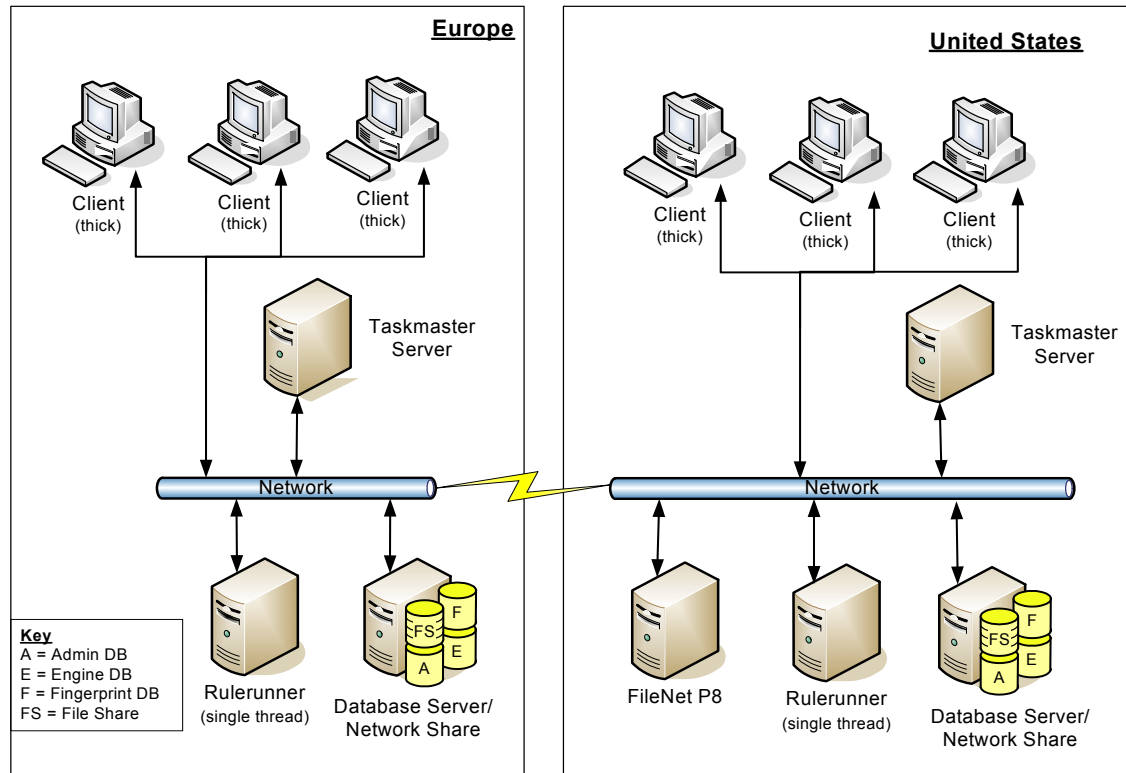


Figure 12-8 Multiple geographies

All the scanned files and batch data are held on a network share drive on the same local area network (LAN) as the users, along with localized Taskmaster Server, Rulerunner servers, and database servers. Only when images and data are ready for committal, they are sent over the wide area network (WAN) to FileNet P8.

Important: The application project files must be identical in both geographies to ensure that both systems work as intended. For more information about these files, which are the `datacap.xml` and `<application>.app` files, see 12.3.2, “Installing a single Rulerunner server” on page 412.

The separation of having two Taskmaster Servers also allows each geography to operate independently of the other, offering a greater degree of resilience if a Taskmaster Server fails.

Alternative configuration for your reference

DISCLAIMER: This alternative option is for your reference only. IBM has not tested nor provides support for this type of configuration. We provide this description strictly for your information.

Another alternative configuration to scale your system across geographies is to run a centralized Taskmaster Server, database server, and file share. However, network bandwidth issues might become apparent when trying to send large quantities of scanned and verification data over the network in remote or low network bandwidth locations.

To overcome this network bandwidth issue, consider using localized network file shares to store batch data. This approach removes the need to repeatedly move files across potentially low-bandwidth network connections. Instead only move them at the point of committal (if required). This way, you can use a centralized Taskmaster Server to manage all batches and localize clients and Rulerunner servers to process the data. Use separate `datacap.xml` files to implement this configuration.

12.2 Rulerunner

Rulerunner can be used in a single-thread and multithread configuration. Such configurations require attention to load balance, race condition, and more as explained in this section.

12.2.1 Single-threaded Rulerunner

With a Production Imaging Edition license, you can run Rulerunner with a single thread for each Rulerunner server, physical machine, or VMWare. By using a single thread, processing of Taskmaster tasks is done in serial, with one process being run at any one time on each Rulerunner server.

Figure 12-9 on page 403 shows the process of a single thread within Rulerunner and how it interacts with the Taskmaster Server, batch folder, Engine database, and Administrator database. This process is repeated each time a task is run by a Rulerunner thread.

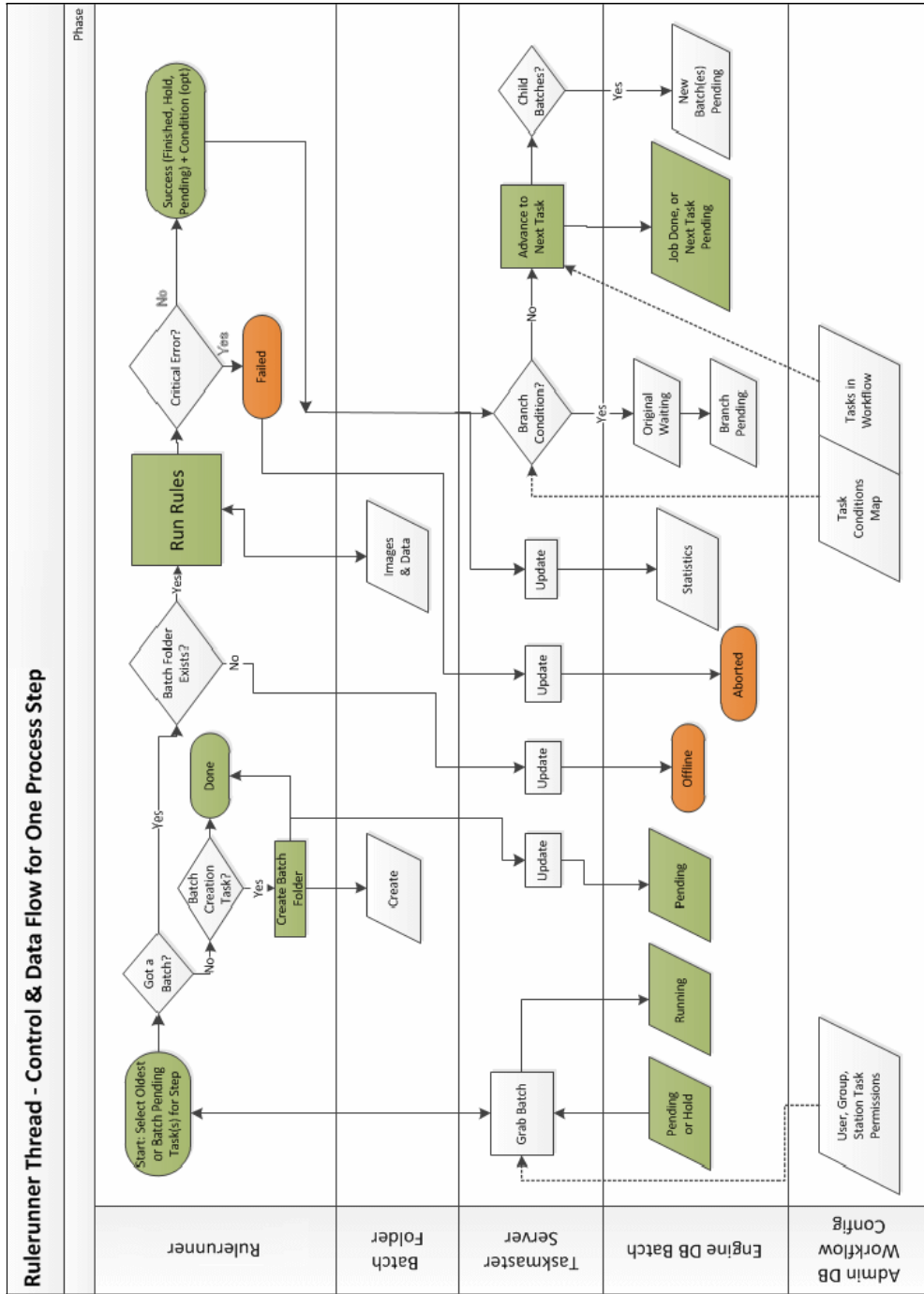


Figure 12-9 Process of a single thread within Rulerunner

The disadvantage of using a single thread is that you do not use the full potential of the hardware that you have available. For example, if you have a quad-core processor capable of running 6 or more threads, you only gain one-sixteenth of the potential throughput of the server.

12.2.2 Multithread Rulerunner

Multithread licensing: At the time of publication, an additional license, called the *Rulerunner Enterprise license*, is required for use of multithreading in Rulerunner. See your IBM marketing representative for more information.

Modern day systems that have multiple processor cores allow for multithreading. To scale up and use this additional processing power, use of multiple threads is advantageous. With a Rulerunner Enterprise license (not currently included with Production Imaging Edition), you can use more than one thread for each physical machine or VMWare.

Rulerunner can manage up to 32 threads. However, an optimal number needs to be determined depending on specific requirements, such as the type of task you are running. If you set the number of threads too high, the system might use up resources, and performance will degrade.

The ratio of threads to processor cores can be a ratio of 1:1, 4:1, or greater. Processing speed and throughput vary depending on the task and must be determined experimentally to ensure optimum performance.

Typically one thread processes a task faster than when four threads process the same task. For example, a Rulerunner server configured to run with only one thread might process a page in 4 seconds. If you now create four threads and process the same task on each one, it might take 5 seconds to process a page for each thread.

If you look at this processing over a minute, in a single-threaded mode, the system processes approximately 15 pages. In a multithreaded mode, the system processes approximately 48 pages. This correlation is not a direct 1:1 relationship. In this example, 15 pages multiplied by 4 equals 60 ($15 \times 4 = 60$), but only 48 pages are processed for the multithread mode with 4 threads. The overall throughput is still higher than a single-thread mode, and you have not increased the footprint of your server room.

Rulerunner is a 32-bit application that can directly access 2–4 GB of memory, depending on the operating system and configuration. However, certain rules create a child process, called *DCOProcessor.exe*, to perform operations such as OCR, barcode reading, and fingerprint creation. Each process runs in its own

32-bit address space. When running under a 64-bit version of the Windows operating system, memory up to 6 GB can be used by the creation processes.

When Rulerunner creates an external process, it monitors the process to detect if any issues occur. A timeout can be configured, after which it considers the external process to be hung. Rulerunner can also be configured to stop its own service if any task stops. A background service detects if Rulerunner stops or hangs and then attempts to restart it. Logging can also be switched on to aid in troubleshooting any issues that might arise.

Figure 12-10 shows a single-threaded Rulerunner server running tasks A, B, C, and D in order on a quad-core server, not using the additional processing capability of those cores. The order the tasks that are run is repeated in a constant loop.

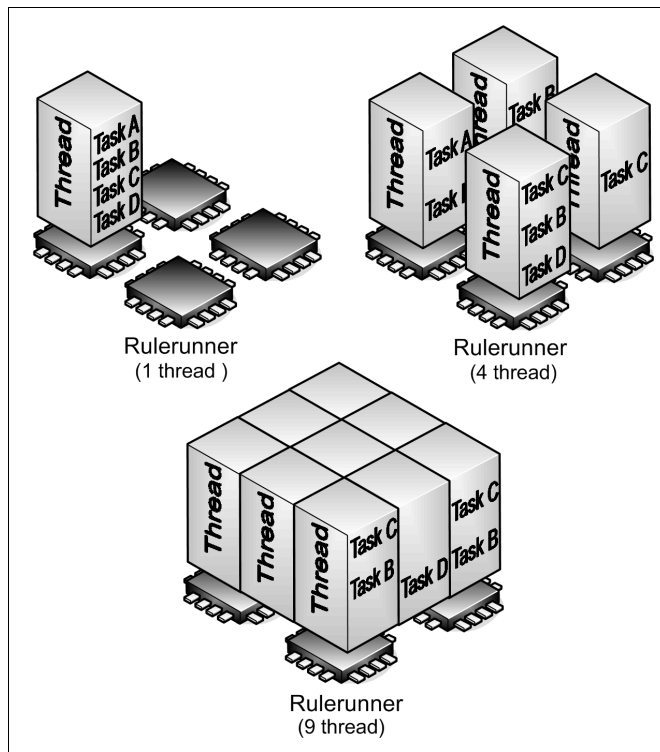


Figure 12-10 Rulerunner threads

The four-threaded configuration shows different tasks being run simultaneously. Some threads are dedicated to a single task, such as task C in the example. Some threads share the load by running different tasks, similar to the single

threaded configuration. The multithread configuration uses the multiple cores that are available on the machine.

The nine-threaded configuration shows different tasks being run simultaneously. Some threads are dedicated to a single task. The configuration also shows that the number of threads is not restricted to the number of processor cores of the server. The number of threads must be optimized depending on the tasks that are being processed.

For a walkthrough of a basic Rulerunner configuration, see 12.3, “Configuring Rulerunner” on page 409. That same section explains how to achieve single threading and multithreading.

12.2.3 Load balancing Rulerunner

Rulerunner can be configured to run in both sequential queuing mode and mixed queuing mode. Choosing the right option is important to scale your system effectively.

Sequential queuing mode

In sequential queuing mode, when all priorities are equal, the order in which the tasks are processed is defined by the order in which they are configured in Rulerunner Manager. The query that is sent to Taskmaster Server consists of only one job for each task pair.

If tasks are given different priorities in Rulerunner Manager, Rulerunner performs the higher priority task on multiple batches before advancing to perform the next task. For example, Thread 1 has Task A with a priority of 8 and Task B with a priority of 2. Priority is calculated based on the lowest common denominator. Therefore, the ratio of 8:2 is simplified to 4:1.

Because Task A is displayed in the thread before Task B, Task A goes first. The thread polls for work from the Taskmaster Server and processes the highest priority, with the oldest available Task A batch available that is pending. It polls four times. Because Task B is displayed next, the thread now polls Taskmaster Server for the highest priority, with the oldest available Task B batch pending. It polls one time.

Mixed queuing mode

In mixed queuing mode, the task priority that is set in Rulerunner is ignored. Taskmaster Server returns the highest priority batches in FIFO order from *all* of the available job and task pairs in the thread. The first batch in this list is processed.

Mixed queuing mode is not appropriate if Batch Creation tasks, such as VScan, are configured with any other tasks.

In mixed queuing mode, Taskmaster always selects the next batch. The position of the tasks in the thread is unimportant.

For an example, see 12.3.4, “Configuring priorities and queuing” on page 425.

12.2.4 Race conditions

When defining which tasks to run on a specific Rulerunner Server or thread, consider the fact that certain actions use the same resources. For example, a task such as VScan polls a defined directory for available files to consume and then creates a batch. If the same task is running in a separate thread or separate Rulerunner Server attempts to run this task, it might encounter issues. Files that are supposed to be consumed in serial order might be broken into separate batches, corrupting the scanning order.

Similarly, a task might require access to a specific file. The first task places a lock on this file until it completes. Other tasks that also require this file therefore must wait until the lock releases before they can continue. This process impacts the performance of subsequent threads that are each waiting for the lock on the file to be released.

When using multiple servers to implement a solution, separate the tasks that might share local resources or that are not thread safe onto different physical machines or virtual machines where possible. This approach helps to avoid any conflicts that might result in an incorrect operation. Also consider how each task or action works, and then test accordingly to minimize any multiprocessing issues.

Important: Consider and test thread safety and multiprocessing when writing custom actions.

12.2.5 Running multithreading in VMware

Rulerunner is supported for use on VMWare. Compared to running it natively, running on VMware results in a drop in performance. Therefore, to achieve optimal performance, while running multiple processes, install Rulerunner onto a native Windows operating system without the virtual machine.

12.2.6 Fingerprint Service

Additional license: At the time of publication, an additional license is required for use of the Fingerprint Service. See your IBM marketing representative for more information.

As explained in previous chapters, Taskmaster uses fingerprinting technology to identify documents as they enter the system. In large-scale Taskmaster implementations, where large numbers of fingerprints (typically over 1000 fingerprints) are required for document identification, use the Fingerprint Service.

The Fingerprint Service overcomes the time-consuming process of loading the fingerprints of the application each time a batch is run. The solution involves reading all the fingerprints into the system memory cache of the Fingerprint Server when the first fingerprint match is requested. To further optimize this process, only the identification portion of the fingerprint is loaded into memory. The fingerprints remain in system memory for the life of the Fingerprint Service process.

When the Fingerprint Service is requested for the first time, it loads all .cco fingerprint files that are defined in the Fingerprint database. The greater the number is of Fingerprints to read into memory, the longer the service takes to start, which can be from a few seconds to a few minutes.

During the startup process, if another Rulerunner workstation tries to call the Fingerprint Service, the second request pauses until the Fingerprint Service completes loading all fingerprints.

If new fingerprints are created on demand by using the Click 'N Key capability and Intellocate functionality, the new fingerprint is loaded into the service. Similarly, if an image that is recognized returns a fingerprint match from the Fingerprint Service that no longer exists in the Fingerprint database, the image is removed from the Fingerprint Service memory. Then the image is re-queried for another fingerprint match. This method negates the potentially lengthy process of reloading the Fingerprint Service when additions or deletions occur.

To use the Fingerprint Service in a project, you use the SetFingerprintWebServiceURL and SetApplicationID actions in the Autodoc Global action library and point them to the server URL that is running the service. For more information, see 12.3.5, “Installing Fingerprint Service” on page 428.

A tool is available to test the Fingerprint Service so that you can add and search for fingerprints and unload the fingerprints of a project if needed.

12.3 Configuring Rulerunner

This section outlines a basic configuration of Rulerunner. It uses the Auto_Claim project developed for the case study in this Redbooks publication. However, this section does not use the same servers.

This section shows a possible setup in a test environment. For further information about setting up in production, see the installation guide that comes with your software.

The servers ECMDEMO1 and ALPHA are used for this installation. ECMDEMO1 is the central server on which all Production Imaging Edition components are installed (Taskmaster and FileNet P8). ALPHA is initially clean and has an additional Rulerunner and an additional Taskmaster Server installed during the following process.

First you see how to configure Rulerunner on a single server where all Datacap components have been installed. Then configuration is done on a separate machine to show how to scale the system both horizontally and vertically.

As mentioned earlier in this chapter, the single server approach is only for demonstration or testing environments.

12.3.1 The Datacap.xml and <project>.app files

To configure the application to run in Rulerunner, use the Taskmaster Application Manager. This application provides a user interface so that you can change Taskmaster applications defined in the `datacap.xml` file.

The `datacap.xml` file contains the centralized Taskmaster Application Service settings that are used by all Taskmaster components in your system. For Rulerunner to run, it requires the location of the `datacap.xml` file by using a file path (the UNC path for separate server installations):

```
C:\datacap\datacap.xml  
\\ECMDEMO1\datacap\datacap.xml
```

Example 12-1 shows a possible configuration of the `datacap.xml` file pointing to various projects on different servers. For Rulerunner to process tasks for a given application, it must be able to gain access the `datacap.xml` file and to the file shares for the applications defined in this file.

Example 12-1 The datacap.xml file

```
<datacap ver="8.0">
<app name="Auto_Claim" ref="//ECMDEM01\Datacap\Auto_Claim"/>
<app name="Flex" ref="//DEM02\Datacap\Flex"/>
<app name="1040ez" ref="//SDEM02\Datacap\1040ez"/>
</datacap>
```

To function, the Taskmaster components must know the location of the `datacap.xml` file. If you use a single server installation, the location is already set at installation time to the `C:\datacap\datacap.xml` directory.

After the application is reached, Taskmaster Application Manager can query the `.app` file in the project folder as follows and as used in this example:

```
C:\Datacap\Auto_Claim\Auto_Claim.app
\\ECMDEM01\Datacap\Auto_Claim\Auto_Claim.app
```

The `.app` file holds all the project paths, connection strings, and other settings that are used by applications to reference the batch folder. Although you can manually alter the `.app` file, use the Taskmaster Application Manager instead. Using Taskmaster Application Manager ensures that XML tags are not omitted and that connections strings for databases are correctly encrypted.

Example 12-2 shows a sample of the `.app` file that is used for Auto_Claim project.

Example 12-2 The auto_claim.app file

```
<app name="Auto_Claim" ver="83" modder="Administrator.ECMDEM01.ECM"
dt="07/08/11.826 10:08:02.826 " src_ver="53">
  <k name="tmserver">
    <k name="tms" ip="ECMDEM01" port="2402" retry="3"/>
  </k>
  <k name="runtime" v="batches"/>
  <k name="tmengine"[encoded].DJj6P7G2gg7Y0BhK7zD52[/encoded]"/>
  <k name="tmadmin" cs="[encoded].DJj6P7GkgFiZ37g5pr5E[/encoded]"/>
  <k name="dco_Auto_Claim">
    <k name="setupdco" v="Auto_Claim.xml"/>
    <k name="rules" v="rules"/>
    <k name="imagefix" v="imagefix.ini"/>
    <k name="UseFPXML" v="False"/>
    <k name="fingerprintconn" cs="[encoded].DJb60zGkY15Y[/encoded]"/>
```

```

    <k name="lookupdb" cs="[encoded].DJj60zGkYVCUTBSe[/encoded]"/>
    <k name="vscanimagedir" v="//ecmdemo1\Datacap\Auto_Claim\image"/>
    <k name="exportdb" cs="[encoded].[/encoded]"/>
  </k>
  <k name="fingerprint" v="fingerprint"/>
  <k name="export" v="export"/>
  <k name="tasks">
    <k name="VScan" profile="VScan"/>
    <k name="PageID" profile="PageID"/>
    <k name="Rulerunner" profile="Rulerunner"/>
    <k name="Export" profile="Export"/>
  </k>
</app>

```

By use of the `datacap.xml` and `<application>.app` files, the location and settings of all projects can be obtained. Use of these files and smart parameters to remove all hardcoded paths and connection strings simplifies promotion from development to production.

Figure 12-11 on page 412 shows the relationship between the `datacap.xml` file, the `.app` files for `Auto_Claim` and `1040ez`, and the subsequent Taskmaster Servers, file servers, and database servers that are used.

The `datacap.xml` file holds the list of applications and where they reside. The program accessing the `datacap.xml` file can use this location. Then it can open the `.app` file to extract the information it needs about the application, such as a database connection string, Taskmaster Servers to use, batch directory location, and so on.

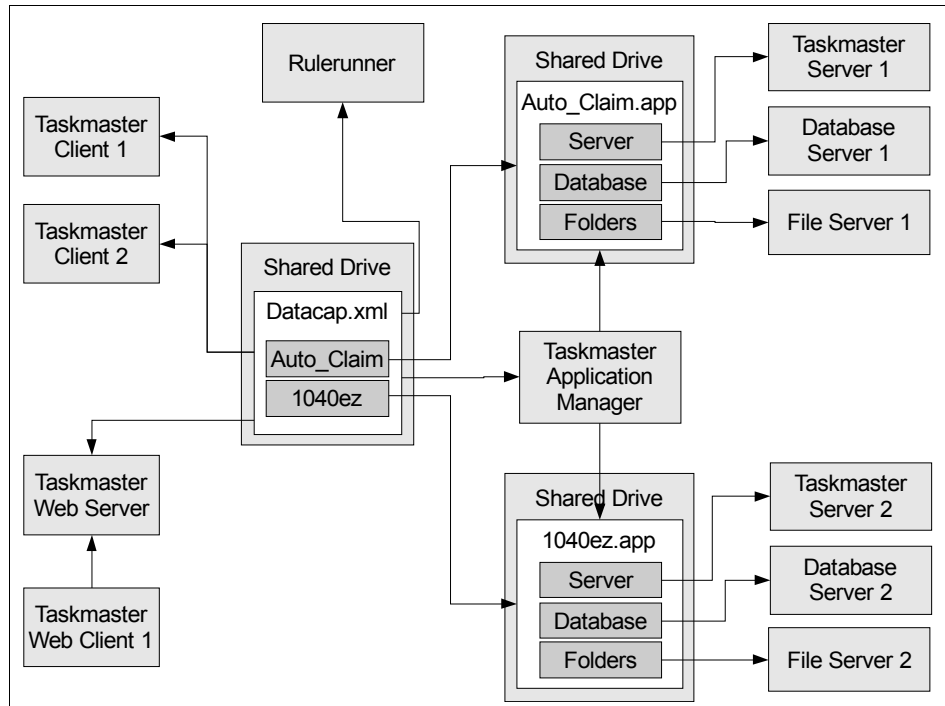


Figure 12-11 The datacap.xml and .app file relationship

12.3.2 Installing a single Rulerunner server

When we carried out the installation of Taskmaster on the single server ECMDEMO1 machine, all Taskmaster components were installed.

On ECMDEMO1, when you open the service console in Windows, you can see the two Rulerunner services as shown in Figure 12-12.

Name ^	Description	Status	Startup Type	Log On As
Datacap Quattro Control Service			Manual	Local System
Datacap Rulerunner Quattro Service			Manual	Local System

Figure 12-12 Rulerunner services on single server installation

In the single server implementation, these services are running under Local System. You are not required to start these services through this console.

Tips:

- ▶ You can start these services through the Windows command console.
- ▶ It is hardly useful to start or stop Rulerunner directly. Therefore, always start or stop the control service by using the **sc /start** and **sc /stop** commands as in the following example:

```
sc \\servername stop "Datacap Quattro Control Service"
```

To install a single Rulerunner server, complete these steps:

1. Selecting **Start** → **All Programs** → **Datacap** → **Taskmaster Client** → **Taskmaster Application Manager**.

The left area of the Taskmaster Application Manager window shows all projects that are currently available on the system. Clicking each project changes the user interface to reflect the locations of key directories, files, and connections.

2. In the Taskmaster Application Manager window (Figure 12-13), click **Auto_Claim** to see all the connections strings, file paths, and other settings associated with the project.

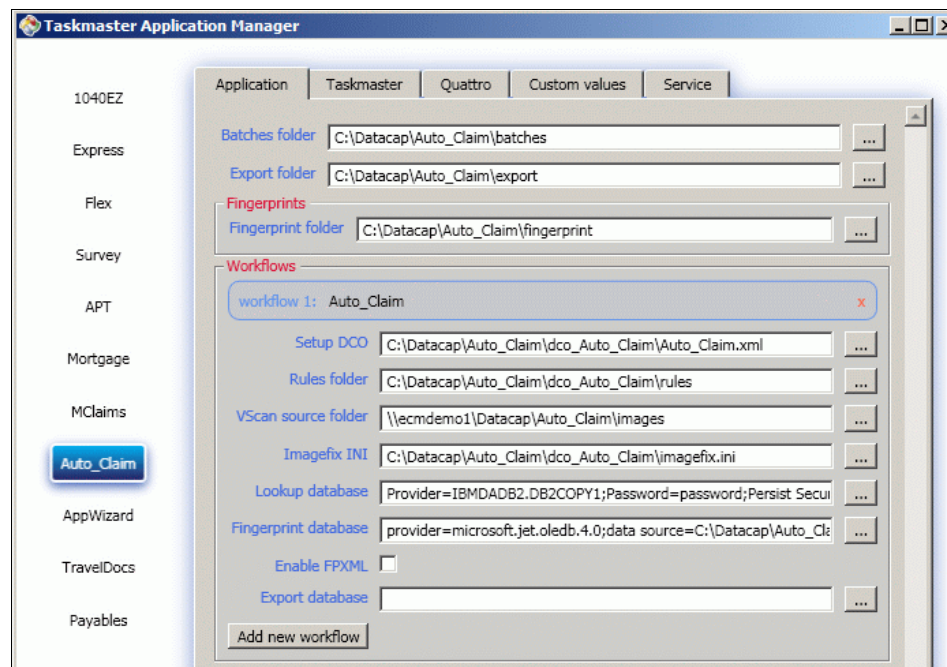


Figure 12-13 Taskmaster Application Manager

3. Click the **Taskmaster** tab (Figure 12-14) to see the server IP and port number. This number is currently set to the localhost IP address of 127.0.0.1, which is acceptable for local installations. Later, when you use a remote server, be sure to change this IP address.

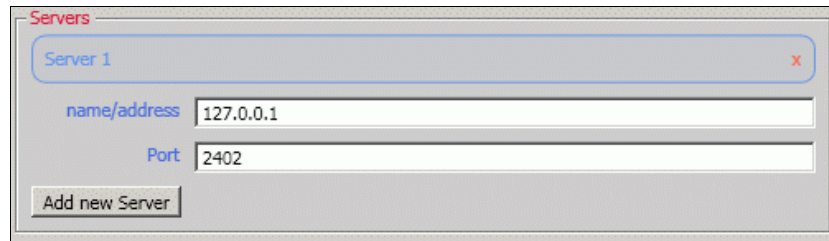


Figure 12-14 Taskmaster Server IP address

4. Create task profiles to run the tasks in the project:
 - a. To find the tasks, in the Taskmaster Client, open the **Auto_Claim** project. Note the names of task as shown on the **Workflow** tab of Taskmaster Administrator (Figure 12-15). In this example, we must run the following tasks:

VScan	Pick up image files from a directory.
PageID	Perform page identification.
Rulerunner	Run the OCR, Validation Rules, and other tasks.
Export	Export to FileNet P8.

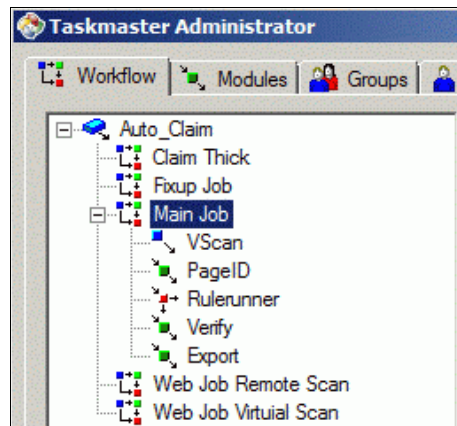


Figure 12-15 Taskmaster Client

- b. Replicate these profiles in the Taskmaster Application Manager user interface for the Auto_Claim project.
- c. Click the **Quattro** tab (Figure 12-16) to ensure that they are spelled correctly with correct usage of uppercase and lowercase.

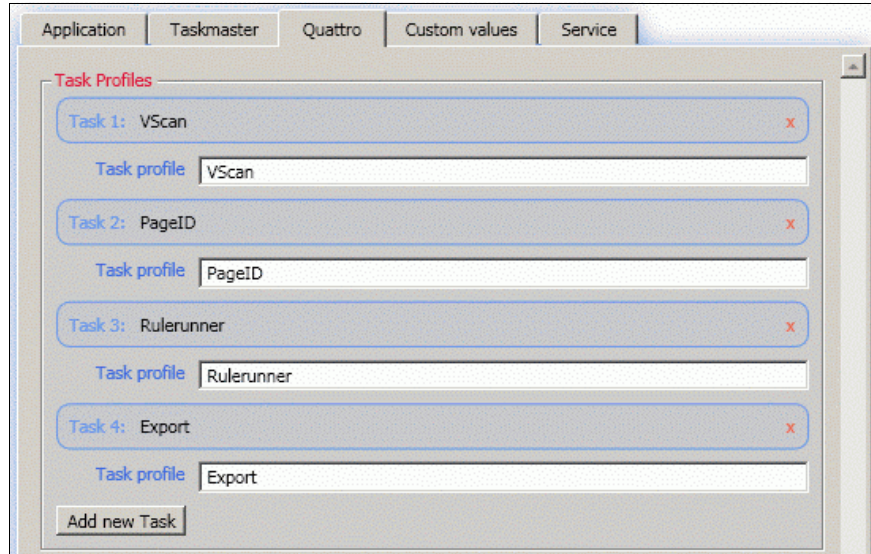


Figure 12-16 Task profiles

- d. Click the **Service** tab (Figure 12-17). Notice that the management path of the main applications points to a local copy of the datacap.xml file. This path is acceptable for a single server installation. When you do a separate server installation, change this path to use UNC.

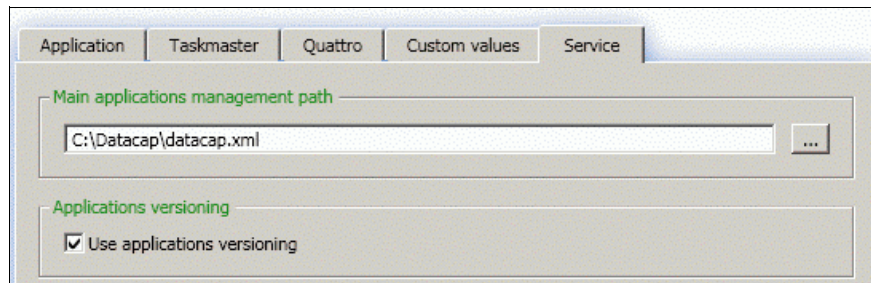


Figure 12-17 Main application management path

- e. Close the Task Application Manager window. All changes are saved when you exit the window.

5. Open the Rulerunner Manager, and then click the **Datacap Login** tab (Figure 12-18).
6. Click **Connect** to connect to the Taskmaster Application Service.

The screenshot shows a window titled 'Datacap Login' with a tab labeled 'Workflow:Job:Task'. The main content area is titled 'Taskmaster Application Service'. Below the title is a large orange 'Connect' button. Underneath the button, there are two radio button options. The first option, 'Taskmaster Authentication. Credentials used to login to Taskmaster.', is selected. Below this option are three text input fields: 'User ID' with the value 'admin', 'Password' with masked characters '*****', and 'Station ID' with the value '1'. The second option, 'Windows Authentication. Credentials used to login to Taskmaster.', is unselected.

Figure 12-18 Datacap login area

When you are prompted for authentication, to get an encrypted password, complete these steps:

- a. Start the **Sit Manager** tool in the C:\Datacap\support\Sit directory.
- b. In the String to Encrypt field, enter your password, and then click the **Encrypt** button. The tool generates an encrypted string for the password.
- c. Enter this string into the Password field on the **Datacap Login** tab. Click **Connect** again for so tha it connects now.

You might be able to use Windows authentication. See the installation documentation for more information.

7. Click the **Workflow:Job:Task** tab. A complete list of the available applications is displayed.
8. Expand **Auto_Claim** to see your tms server and databases. You see a graphical view of the available Taskmaster Server and databases.
 - a. Select each database.
 - b. Right-click **tms** and select **Connect**.
 - c. Repeat these steps for the databases.

You see the connection status change at the bottom of the window. The available tasks in the tree look similar to the tasks shown in the left pane of Figure 12-19.

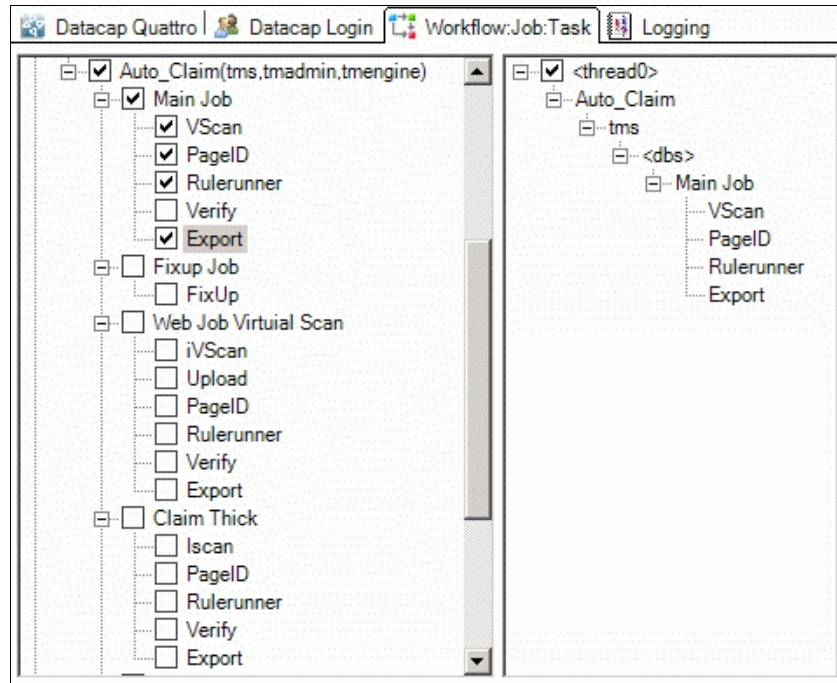


Figure 12-19 Workflow, Jobs, Tasks

At this point, you can run Rulerunner in either single-threaded mode or multithreaded mode. Currently an additional license, which is the *Rulerunner Enterprise license*, is required to allow use of multiple threads. Figure 12-19 shows a single-threaded mod example.

9. Right-click the blank right pane, and add three threads (license permitting). Select and then drag the tasks from Main Job to each thread, which creates three separate threads that run simultaneously (Figure 12-20).

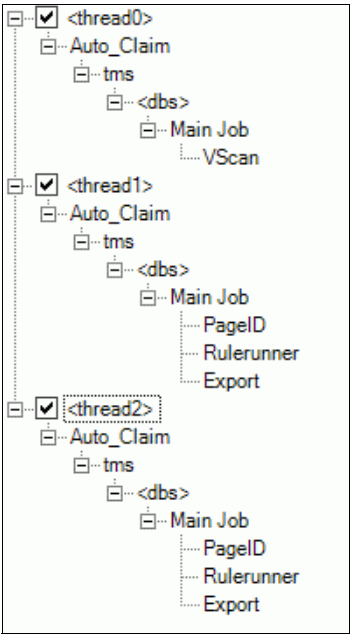


Figure 12-20 Multiple threads

As stated earlier in this chapter, a limit of 32 threads is available. However, as resources are consumed, degradation of performance usually occurs before you reach this limit.

10. Optional: Configure priorities in which the tasks run by using the following options:
- Priority
 - Skip Same Batch
 - Mixed Queuing

Figure 12-21 shows an example of how this priority is configured.

ID	
name	Rulerunner
Settings	
priority	1
skipsamebatch	0

Figure 12-21 Priority configuration

For more information about priorities, see 12.3.4, “Configuring priorities and queuing” on page 425.

11. Optional: Click the **Logging** tab (Figure 12-22) to turn on logging. Files generated for each thread being run are created in the C:\Datacap\tmc\lient directory of the Rulerunner server.

The screenshot shows the 'Logging' tab in the Datacap Quattro application. The window title bar includes 'Datacap Quattro', 'Datacap Login', 'Workflow:Job:Task', and 'Logging'. The main content area has a yellow background and contains two sliders and a list of checkboxes. The first slider, labeled 'Level of detail written to the RRS logs.', ranges from 'None' to 'All' with a slider knob positioned near 'All'. Below it are seven checked checkboxes: 'Batch Log', 'Log Application ID', 'Log Override', 'Log Reflush', 'Log Severity', 'Log Threadproc', and 'Log Time'. The second slider, labeled 'Severity level of messages logged.', also ranges from 'None' to 'All' with a slider knob positioned near 'All'. At the bottom, there is a 'Quick info' section with a text box containing the text 'How much information should RRS output to it's own log'. The bottom of the window has a tab bar with 'System Event Log', 'Quattro Log', and 'RRS Log'.

Figure 12-22 Logging in Rulerunner

12. Click **Save** to save this thread configuration.
13. Disconnect from the Taskmaster Application Service. Click the **Datacap Login** tab, and then click **Disconnect**.

The settings that were saved are written to both the system registry and an XML file that by default is in the C:\Datacap\tmc\lient directory.

14. Optional: Create multiple configuration files with different names and load them manually. On the **Workflow.Job.Task** tab (Figure 12-23), change the value in the Path to Settings File field.

Thread Timeout: 1800 Number of seconds before threads are forced to stop.

Sleep For: 3 Number of seconds each thread idles.

Path to Settings File: C:\Datacap\tmlclient\Quattro.xml

Figure 12-23 Rulerunner XML file

With these different configurations, you can change the processing order, priority, and so on, quickly to predefined configurations by stopping Rulerunner and pointing it to the subsequent `quattro.xml` configuration file. These settings are ideal when processing a given task requires priority over others, such as clearing a back log.

15. Ensure that no outstanding tasks remain in the Taskmaster Client Job Monitor.
16. Close the Taskmaster Client.

Important: A client that is open on the same machine where Rulerunner is running can cause issues with connections.

17. Click the **Datacap Quattro** tab in Rulerunner Manager and click **Start** to start Rulerunner.

Rulerunner now starts and begin to run the tasks that are assigned to it. No view is available to indicate that it is working. However, you can go to the project batch folder in the `C:\Datacap\Auto_Claim\batches` directory and see if file creation or update activity is occurring. Alternatively open the Taskmaster Client, and see the tasks processing in the Job Monitor window. Figure 12-24 shows an example of four threads running `PageID`.

83	110708.017	Main Job.PageID	running	7/8/2011 6:40:50	7/8/2011 6:48:0
82	110708.016	Main Job.PageID	running	7/8/2011 6:40:50	7/8/2011 6:48:0
81	110708.015	Main Job.PageID	running	7/8/2011 6:40:50	7/8/2011 6:48:0
80	110708.014	Main Job.PageID	running	7/8/2011 6:40:51	7/8/2011 6:48:0

Figure 12-24 Tasks shown in Taskmaster Client Job Monitor

18. As a rest, run a VScan-based task to consume files from a folder. If this task is set to repeatedly grab these files and not delete them, you see batches begin to be generated in the project batch folder.

If you open Windows Task Manager, you see several Document Hierarchy (DCO) processes running in the list.

You can also check for log files if you turned up the logging levels and defined a specific location for them. The log files are useful for debugging issues that might occur. High levels of logging can affect performance. Therefore, it is advantageous to drop them to a suitable level depending on the environment on which you are running.

12.3.3 Installing an additional Rulerunner server

Tip: Before you read this section, read 12.3.2, “Installing a single Rulerunner server” on page 412, to help you understand the terminology used in this section.

Running Rulerunner on a separate machine requires additional setup. For our example, we use a build of Windows 2008 R2 64-bit. We show a possible setup in a test environment. For more information about setting up the production environment, see the installation guide that comes with your software.

For the purposes of this procedure, we refer to the new machine as ALPHA and the original machine as ECMDEMO1.

Windows is running in a domain which is a requirement for Rulerunner and all Taskmaster software components. All software on all machines must be in the same domain.

To install an additional Rulerunner server, complete these steps:

1. Enable the domain:
 - a. Select **Start** → **Control Panel** → **System** → **Change Settings**.
 - b. Click the **Computer Name** tab, and then click **Change**.
 - c. Enter the domain name you want to join.
 - d. After join the domain, reboot Windows.
 - e. Log back in to ALPHA with the domain Administrator account.
2. Create a domain user to run the Rulerunner service:
 - a. On the Domain Controller server, which is ECMDEMO1 in this example, select **Start** → **Administrative Tools** → **Active Directory Users and Computers**.
 - b. Enter an appropriate name for the user.

For this example, we create the user *QuattroAdmin*. Make this user a member of the Domain Admins group to give appropriate privileges to run Rulerunner. Figure 12-25 shows the setting.

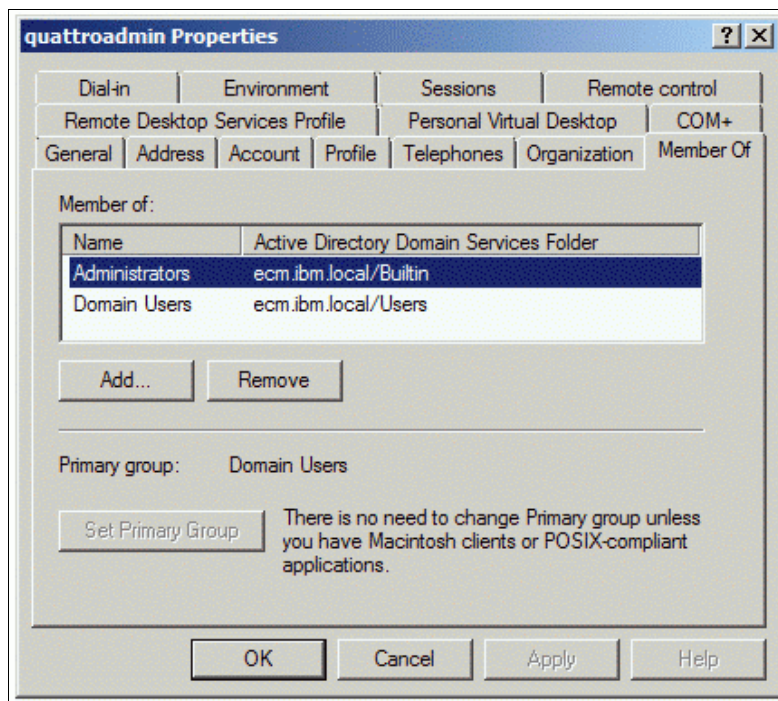


Figure 12-25 Rulerunneradmin group memberships

If you want to try a different approach because security is an issue, see the installation documentation.

3. Allow Rulerunner to access the application files on the Taskmaster Server:
 - a. Share the C:\Datacap\Auto_Claim folder with access permitted to Rulerunner admin.
 - b. To complete the connection, set up a network share on ALPHA to the shared C:\Datacap\Auto_Claim folder on the ECMDEMO1 machine.
4. Install Rulerunner on ALPHA:
 - a. Double-click the Datacap install executable. Do not install all components, select only **Rulerunner/Rulerunner Enterprise**.
 - b. After the installation is complete, restart ALPHA to make any changes take effect.
5. Open Taskmaster Application Manager on ALPHA.

- Click the **Service** tab (Figure 12-26). Then change the Main applications management path value to point to the shared network folder where the `datacap.xml` file resides, which is `\\ecmdemo1\datacap\datacap.xml`. This XML file contains details about the current projects.

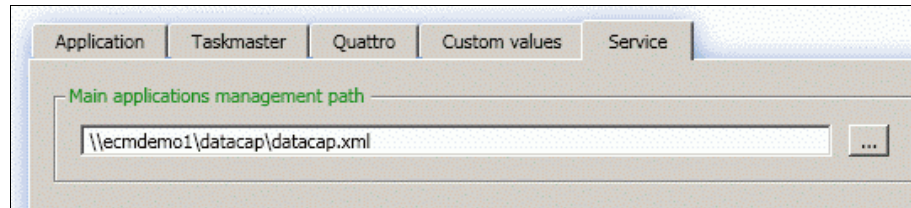


Figure 12-26 Service path

The Taskmaster Application Manager user interface is updated with details of the projects. The location of the required files and directories now uses UNC paths as shown in Figure 12-27.

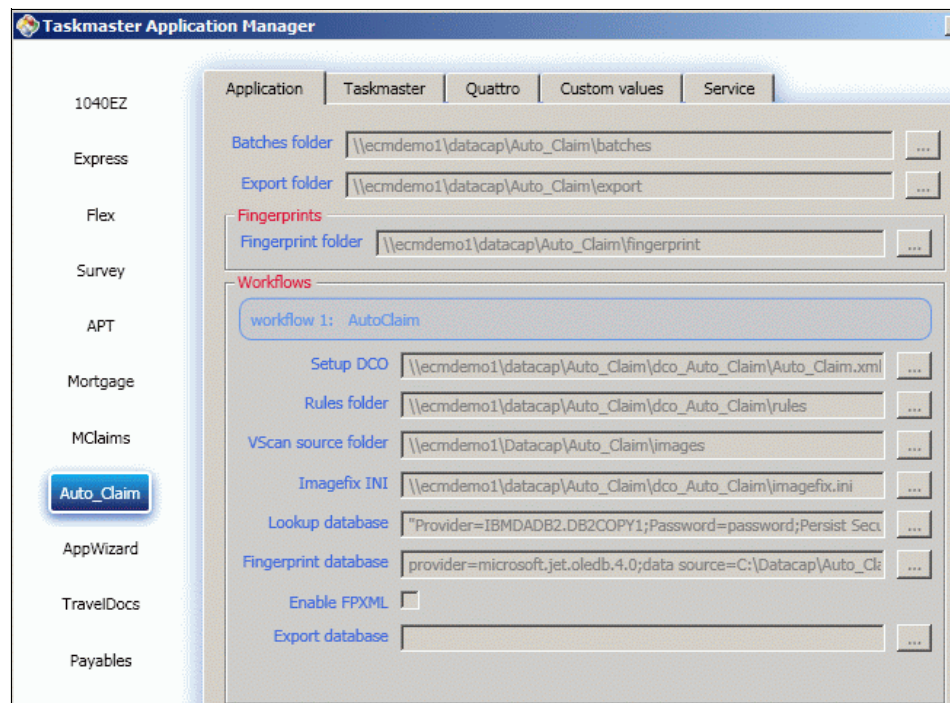


Figure 12-27 Taskmaster Application Manager

- Close the Taskmaster Application Manager for the changes to take effect.

8. Open the Taskmaster Application Manager on ECMDEMO1. Notice the difference because these settings are not configured using UNC paths (as seen on ALPHA).
9. Click the **Taskmaster** tab. Change the Taskmaster Server name from the localhost address of 127.0.0.1 to the server name of ECMDEMO1. This setting allows nonlocal machines to connect to the Taskmaster Server.
10. Close Taskmaster Application Manager.
11. Go back to ALPHA, and then open its Taskmaster Application Manager.
On the **Taskmaster** tab, you now see the server name ECMDEMO1 as opposed to 127.0.0.1.
12. Set the VScan string to a UNC path. This string is not automatically changed like the other variables because you might want to point it to a different file system. See Figure 12-27 on page 423 for an example.
13. Change the user that runs the Rulerunner services to the recently created QuattroAdmin user:
 - a. Open the service console.
 - b. Change Log on as user to the full domain user ID (quattroadmin@ecm.ibm.local), and enter the appropriate password.
 - c. Start the service from the Services window to ensure that the user name and password work as intended. Do not start the service this way normally.
 - d. If the user credentials work, ensure that the services are stopped (Figure 12-28).



Name ^	Description	Status	Startup Type	Log On As
 Datacap Quattro Control Service		Started	Manual	quattroadmin@ecm.ibm.local
 Datacap Rulerunner Quattro Service			Manual	quattroadmin@ecm.ibm.local

Figure 12-28 Rulerunner server services running as quattroadmin@ecm.ibm.local

14. Open the Rulerunner Configuration Console.
15. Log in by using the Taskmaster Service user name and password.
16. When you are prompted about encryption, use the Sit Manager from the C:\Datacap\Support\sit directory to access the encrypted password string.
17. After the connection is made, click the **Workflow:Job:Task** tab, and expand the **Auto_Claim** folder. Then, as in 12.3.2, “Installing a single Rulerunner server” on page 412, drag one of the tasks to the thread. It populates the thread with details of the task.
18. Click **Save** to commit your changes to the quattro.xml configuration file.

- 19.Optional: Click the **Logging** tab, and turn on logging as explained previously.
- 20.Click the **Datacap Login** tab to disconnect from Taskmaster Services.
- 21.Click the **Datacap Quattro** tab, and then click **Start** to start the Rulerunner server.

Rulerunner now starts and processes any tasks that are assigned to it. As mentioned previously, a visual indication to show that it is working is not available. Look in the project batch folder in the C:\Datacap\Auto_Claim\batches directory on ECMDEMO1. This folder shows an indication of processing occurring. Alternatively open the Taskmaster Client, and then view the tasks processing in the Job Monitor window.

Test: A good test is to run a VScan-based task to consume files from a folder. If VScan is set to repeatedly grab these files and not delete them, you see batches begin to be generated in the project batch folder.

If you open Windows Task Manager, you see several DCO processes in the window as shown in Figure 12-29.

Image Name ^	User Name	CPU	Memory (Private Working Set)	Description
DCOProcessor.exe *32	SYSTEM	24	54,560 K	Generic DCO processor
DCOProcessor.exe *32	SYSTEM	25	54,656 K	Generic DCO processor
DCOProcessor.exe *32	SYSTEM	25	54,868 K	Generic DCO processor

Figure 12-29 Three DCOProcessors running on Windows

12.3.4 Configuring priorities and queuing

As mentioned previously in this chapter, you can configure priorities against specific tasks, jobs, and so on. You can configure priorities and queuing, for example, in the Rulerunner Manager as shown in Figure 12-30.

ID

name

Rulerunner

Settings

priority

1

skipsamebatch

0

Figure 12-30 Configuring priority and skipsamebatch

If we open the `quattro.xml` file in the `C:\Datacap\tmlclient` directory, we can see in more detail the parameters that are available for each thread, application, server, database, job, and task (Example 12-3).

Only use this file as a reference. Do not change any of the values in the file. Instead, use the Rulerunner Manager to change the values.

Example 12-3 Quattro.xml file

```
<thread0 enabled="1">
  <app name="Auto_Claim" priority="1">
    <server name="local" priority="1">
      <db admin="tadmin" engine="tmengine" priority="1" >
        <job name="Main Job" priority="1">
          <task name="Batch Profiler" skipsamebatch="0" priority="1"
        />
          <task name="VScan" skipsamebatch="0" priority="3" />
        </job>
      </db>
    </server>
  </app>
  <app name="1040ez" priority="2">
    <server name="local" priority="1">
      <db admin="tadmin" engine="tmengine" priority="1" >
        <job name="Demo" priority="1">
          <task name="VScan" skipsamebatch="0" priority="1" />
        </job>
      </db>
    </server>
  </app>
</thread0>
```

Look at the following important parameters in the XML file:

SkipSameBatch	Is used when a task might set the batch back to a pending state. If you use VScan configured to remove images after creating the batch, when VScan runs again, and no images are available, the batch is reset to pending. To prevent running the batch repeatedly, this option introduces a delay. Therefore, if the Batch ID is the same next time it runs VScan, do not do anything for X seconds, effectively stopping a repeated loop until images are available.
Priority	Is used to determine the ratio that the current thread needs to spend processing the current node among nodes of the same level. The priority is only applicable

when more than one node exists at the same level. For example, more than one application (app) exists under one thread or multiple tasks (task) under one job.

The following queuing modes determine how the priority value is used:

- | | |
|-------------------|---|
| Mixed | Priorities that are below database (dbs) level are ignored by Rulerunner. Priorities below the (dbs) level are determined by the Taskmaster Server. |
| Sequential | The combined value of priorities of all levels is used to determine how often each particular job and task pair must be run. |

The priorities are used to calculate a ratio. All the priorities are divided on the smallest value and rounded. For example, if we set priorities of 5 and 2, they are divided by a ratio of 3:1.

Thread-time distribution is based on the highest level node first. For example, multiple applications are specified, and the ratio is 1:2. If any priorities are set on the task level, the real ratio is combined with the higher level.

Example 12-3 on page 426 has two applications. thread0 queries batches from the 1040ez application twice more than it queries batches from the Auto_Claim application. The Auto_Claim application also has multiple tasks specified. These tasks depend on the queuing mode used:

- ▶ In mixed mode, it queries the Taskmaster Server for the batch with highest priority among pending batches for Main Job/Batch Profiler and Main Job/VScan job and task pairs.
- ▶ In sequential mode, each time the thread queries the Taskmaster Server, it first tries grabbing a batch for VScan for three times. Only on the fourth attempt, the Batch Profiler batch is grabbed.

In sequential mode, if you have an unlimited amount of batches for all job and tasks pairs from all applications pending it does the following processing:

1. Runs 1040ez VScan two times.
2. Runs Auto_Claim VScan one time.
3. Runs 1040ez VScan two times.
4. Runs Auto_Claim VScan one time.
5. Runs 1040ez VScan two times.
6. Runs Auto_Claim VScan one time.
7. Runs 1040ez VScan two times.
8. Runs Auto_Claim Batch Profiler one time.

If no batches are pending for any job and task pair, this job and task pair are queued next time according to the following formula:

$$s = 2^n;$$

where:

s Delay in seconds.

n Amount of unsuccessful attempts to grab the batch.

The maximum wait time will not exceed 64 seconds.

The sample set of values might look like the following process, where each step is an unsuccessful attempt to grab a batch:

1. Wait for 2 seconds
2. Wait for 4 seconds
3. Wait for 8 seconds
4. Wait for 16 seconds
5. Wait for 32 seconds
6. Wait for 64 seconds
7. Wait for 64 seconds
8. Successfully grabbed batch
9. Wait for 2 seconds
10. Wait for 4 seconds
- 11....

If all job and task pairs have no pending batches, the thread goes to sleep for the time interval specified in the registry (default is 10 seconds).

12.3.5 Installing Fingerprint Service

The Fingerprint Service is a web service that runs on IIS. Its purpose is to aid in reducing fingerprint loading times on large-scale implementations using large numbers of fingerprints. For more information, see 12.2.6, “Fingerprint Service” on page 408.

This section shows a possible setup in a test environment. For further information about setting up in production, see the installation guide that comes with your software.

Additional license: At the time of publication, an additional license is required for use of the Fingerprint Service. Consult your IBM marketing representative for more details.

To install the Fingerprint Service, complete the following steps in which we use a build of Windows 2008 R2 64-bit.

1. Run the installation of Datacap on the machine on which you want to run the Fingerprint Service. The **RulerunnerServer** component is required during installation.
2. After the installation is complete, go to the C:\Datacap\FingerprintService directory. Here, you see the web.config file and other required folders and domain link library (DLL) files.
3. Click **Start** → **Control Panel** → **Programs and Features**.
4. Turn the Windows features on or off.
5. Open the Server Manager and click **Server Manager**, expand **Roles**, and click **Webserver (IIS)**. If it is not installed, consult the Microsoft documentation to install it.
6. Right-click **Webserver (IIS)** and select **Add Role Services**. Ensure that the following options are selected:
 - Add Webserver
 - Add Common HTTP features
 - Add Application Development (ensuring that ASP.Net and ASP are selected)

Figure 12-31 shows the setup.

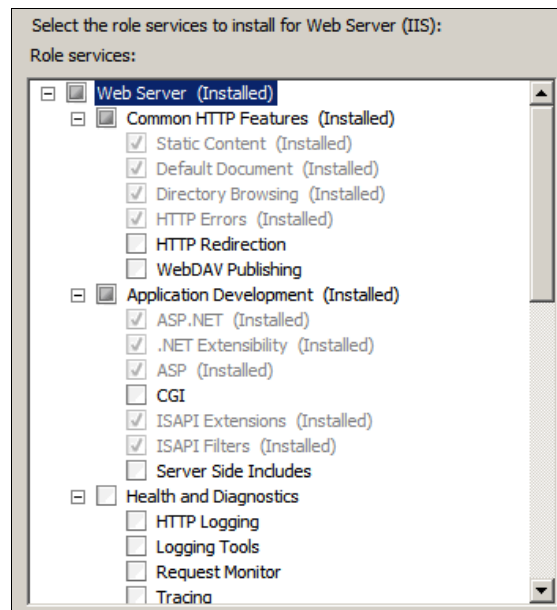


Figure 12-31 Role Services for IIS

7. Install the components, and then reboot the server.
8. In Windows 2008 R2, open **Server Manager**, expand **Roles** → **Web Server** → **Internet Information Services**.
9. In the Connections pane:
 - a. Expand the tree. Right-click **Application Pools**, and then select **Add Application Pool**.
 - b. For the application pool name, enter `FingerprintService`. As the framework version, select **.NET Framework v2.0.50727**.
 - c. Select **Integrated** as the managed pipeline. Ensure that **Start application pool immediately** is selected. Then click **OK**.
10. In the Connections pane, complete these steps:
 - a. Expand the tree. Right-click **Default Web Site**, and select **Add Application**.
 - b. For Alias, enter `FPService`, and then for Application pool, select **FingerprintService**. Click **OK**.
 - c. Click the **browse** button, and then navigate to the installation directory for the fingerprint service. By default, the directory is `c:\Datacap\FingerprintService` for the physical path.
 - d. Click **Connect As**, and then select the **Specific user** radio button.
 - e. Click **Set**, and then enter the information for the desired user account. For this installation, we use Administrator. In practice, this account must have read and write permissions to this registry branch and sub keys (`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog`). For the purposes of this book, we use Administrator, although you might want to change it. You might also want to make it part of the domain where the Taskmaster and Rulerunner servers reside.
 - f. Click **OK** to close the open dialog boxes.
11. From the connection pane, complete these steps:
 - a. Click **Application Pools**, and then select **FingerprintService**.
 - b. In the Actions pane, click **Advanced Settings**, and then set Enable 32-Bit Applications to **True**.
 - c. Change Idle Time-out (minutes) to 0. Change Identity to the account that you configured for the fingerprint service application (in this case administrator). Then click **OK**.
12. Open a web browser and point to the following URL:
`http://localhost/FPService/service.asmx?WSDL`
If successful, you see a web service WSDL page.

13. To test the Fingerprint Service, complete these steps:

- a. Select **Start** → **All Programs** → **Datacap** → **Support** → **Fingerprint Service** to open the Try Fingerprint Service.

Figure 12-32 shows the Try Fingerprint Service tool window. With this tool, you can test the service by uploading single .cco images or the contents of the entire directory. Then after they are loaded, you can query it for a match by using a specific .cco file. You can also extract statistics and other information from the service.

- b. Ensure that you have the correct server name entered into the URL path, which is `http://localhost/FPService/service.asmx?WSDL`. You can also enter this path into a browser to view the WSDL.
- c. Click **First 'N' Records**, which returns “No Fingerprints loaded” in the right pane.
- d. For the Upload Fingerprint field, click the ... button, and then look in the `C:\Datacap\Flex\fingerprint` directory for a single .cco file. Click **Upload Fingerprint**.
- e. For the Find Fingerprint field, click the ... button, and then look in the `C:\Datacap\Flex\fingerprint` directory for the same .cco file. Click **Find Fingerprint**.

You now see that a match has been made.

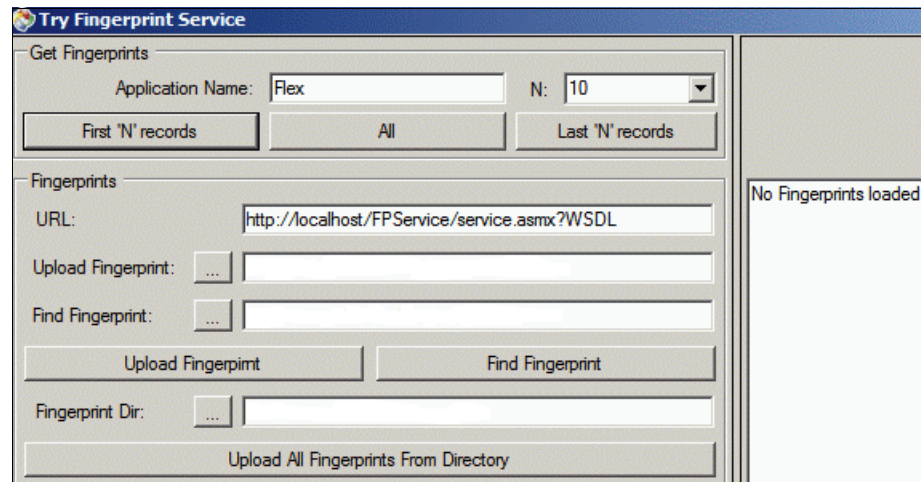


Figure 12-32 Try Fingerprint Service

You can experiment with loading entire directories of .cco files and matching individual fingerprints, for example.

To use this service in a production environment, use a dedicated server to install and run this service.

12.3.6 Using the Fingerprint Service

To use the Fingerprint service, you specify in the application where the web service resides, which is a simple process that you can easily test.

You need to add the following additional actions to your project:

SetFingerprintWebServiceURL

Tells the application where the web service resides. This action shows that web service is held locally, although it is run in a test environment.

SetApplicationID

Sets the application ID in the Fingerprint service so that fingerprints can be added and removed only from that application.

Figure 12-33 shows an example of these actions used in a project.

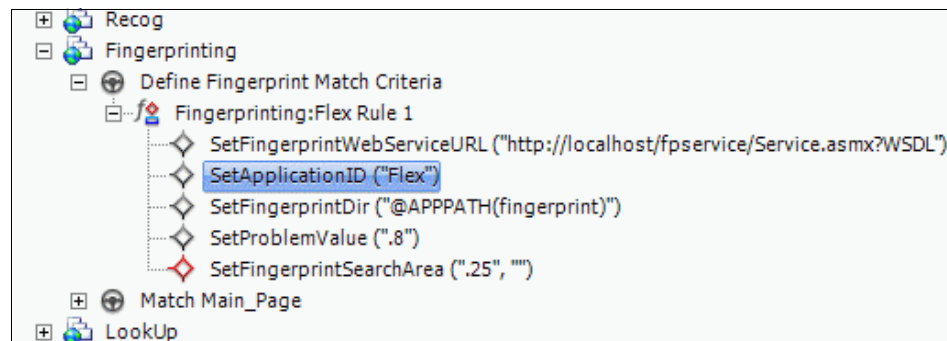


Figure 12-33 Fingerprinting setup used in a project

After you run these actions, you see a log similar to Example 12-4.

Example 12-4 Fingerprint run result

```
----- FindFingerprint of:
c:\datacap\flex\batches\20110193.002\tm000001.tif -----
Updating SearchArea. Top: 0; Bottom: 0.25
Loading Fingerprints...
Fingerprint Record Count: 5
FP Service flag: True
0 fingerprints already loaded...
Updating CCO Fingerprint Service...
```



```
Loading FP Service
FPLoading station 1
Attempting to load Fingerprints
Loaded c:\datacap\flex\fingerprint\555.cco
Loaded c:\datacap\flex\fingerprint\1301.cco
Loaded c:\datacap\flex\fingerprint\1302.cco
Loaded c:\datacap\flex\fingerprint\1303.cco
Loaded c:\datacap\flex\fingerprint\1304.cco
Cleared the FPLoading flag
Loading time: 0.1875
Matching with the Fingerprint Webservice...
strCCOFile: c:\datacap\flex\batches\20110193.002\tm000001.cco
m_SearchArea: 1
sRetn: 0.96,2,0,0
Confidence: 0.96
Template Index: 2
xOffset: 0
YOffset: 0
The FP Service returned fingerprint: 1301
Fingerprint: 1301; index = 2; Confidence = 0.96
Match existing Fingerprint: 1301; Image:
c:\datacap\flex\batches\20110193.002\tm000001.tif
Fingerprint Matching Time: 0.1719
t:C28 p:48E2878 FindFingerprint returns True
```

This log shows the fingerprints that are being loaded to the web service and the matching that is occurring.

On larger scale implementations, subsequent runs of this fingerprinting process is faster, negating the need to load all fingerprints for every batch.

After you run the Fingerprint Service for the first time, use the Try Fingerprint Service too, which you can access by selecting **Start → All Programs → Datacap → Support → Fingerprint Service**.

By using this tool, you can enter the application ID that is used in the application (defined in **SetApplicationID**) and obtain information about the fingerprints. Figure 12-34 shows an example.

Figure 12-34 Try Fingerprint Service returning fingerprints added by an application

12.4 Adding additional Taskmaster Servers

Similar to how you can add additional Rulerunner servers with relative ease, you can add additional Taskmaster Servers. This section shows how to configure two Taskmaster Servers so that they work in an active-active mode to create additional processing power and add redundancy. This section shows a possible setup in a test environment. For more information about setting up in production, see the installation guide that comes with your software.

12.4.1 Adding a failover Taskmaster Server

This section builds upon the configuration in 12.2.2, “Multithread Rulerunner” on page 404. This configuration consists of a central server, named ECMDEMO1, with all Datacap components on it, and a secondary server, named ALPHA, which currently only has Rulerunner. Both ECMDEMO1 and ALPHA are running in the same domain.

To add a failover Taskmaster Server (in our example ALPHA), complete these steps:

1. Install Taskmaster Server on the second server, ALPHA. Follow the standard Datacap installation procedure. Choose the Taskmaster Server module and the client module (to test the setup).
2. After completing the installation, restart the machine.
3. Enable the domain on ALPHA.

Windows must be running in a domain, which is required for Taskmaster Server and all Taskmasters software components. All software on all machines must be in the same domain.

To enable the domain on ALPHA, complete these steps:

- a. Select **Start** → **Control Panel** → **System** → **Change Settings**.
 - b. Click the **Computer Name** tab, and then click **Change**.
 - c. Enter the domain name of the failover server that needs to join.
 - d. After the failover server joins the domain, restart Windows.
 - e. Log in again with the domain Administrator account.
4. Create a domain user under which to run the Taskmaster Server service:
 - a. On the Domain Controller server, which is ECMDEMO1 in this example, select **Start** → **Administrative Tools** → **Active Directory Users and Computers**.
 - b. Enter an appropriate name for the user. For this example, we enter TM Admin as the user. Make this user a member of the Domain Admins group to give appropriate privileges to run Taskmaster Server.

For more information about using a different approach if security is an issue, see the installation documentation.

Tip: When trying to connect to the Taskmaster databases later, error messages might result because you did not configure the tadmin user correctly. The new user might not have the permissions to access the shared directory or databases. Check to ensure that the user has that permission.

5. Add the information of the second server, ALPHA, to the first server, ECMDEMO:
 - a. On the first server, ECMDEMO1, where the datacap.xml file is located, open **Taskmaster Application Manager**.
 - b. Select the project, which in this example is **Auto_Claim**, and then click the **Taskmaster** tab.
 - c. From the **Taskmaster** tab, inform the system where additional Taskmaster Servers are located. Figure 12-35 shows that only one Taskmaster Server is currently configured.

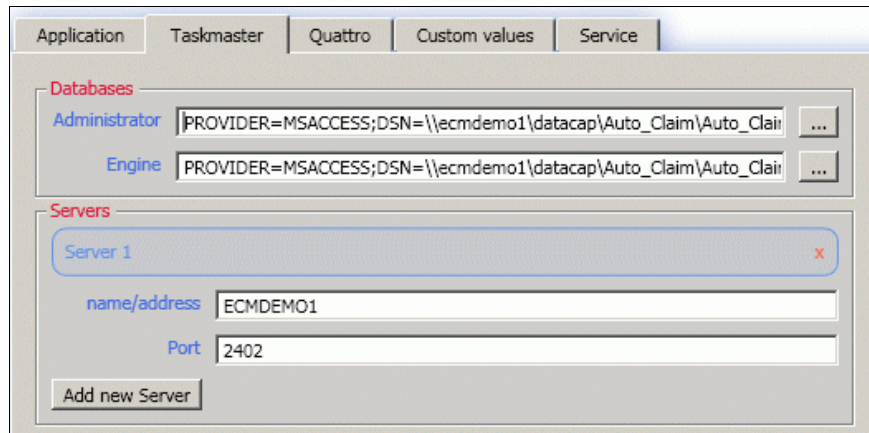


Figure 12-35 One Taskmaster Server configured in Taskmaster Application Manager

- d. Click **Add new Server**, and then enter the server name or the IP address and port number for the second server. For our example, we enter a server name of ALPHA and a port number of 2402.
- e. Close the Taskmaster Application Manager to save these changes.

Figure 12-36 shows the two Taskmaster Servers that are configured.

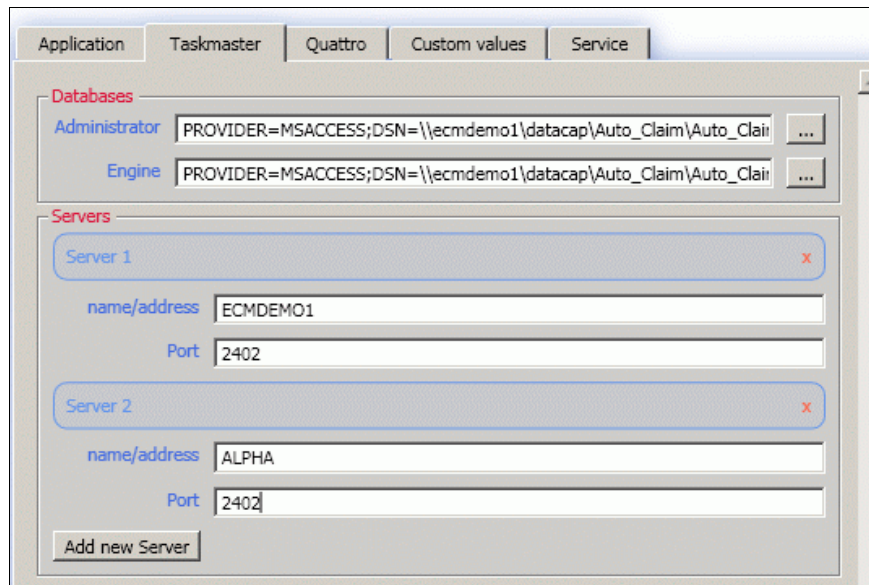


Figure 12-36 Two Taskmaster Servers configured

- f. Open the c:\Datacap\Auto_Claim\Auto_Claim.app file or an .app file that you have been working with. You see two lines of XML code similar to the lines in Example 12-5.

Example 12-5 Original Auto_Claim.app file

```
<k name="tms" ip="ECMDEMO1" port="2402" retry="3"/>
<k name="tms" ip="ALPHA" port="2402" />
```

- g. Add retry="3" to the second Taskmaster Server entry line to ensure that Rulerunner can connect. Example 12-6 shows the updated line.

Example 12-6 Updated Auto_Claim.app file

```
<k name="tms" ip="ECMDEMO1" port="2402" retry="3"/>
<k name="tms" ip="ALPHA" port="2402" retry="3"/>
```

- h. Go to the ALPHA server where the second Taskmaster Server is installed, and then open **Taskmaster Application Manager**.
- i. Click the **Services** tab. Enter the shared drive location of the datacap.xml file if it is not already present. In our example, the location is \\ecmdemo1\datacap\datacap.xml.

- j. Refresh the user interface to show a list of the available projects (as defined in the `datacap.xml` file).
 - k. Click the **Taskmaster** tab. You see the two servers: the primary Taskmaster Server, called ECMDEMO1, and the second server, called ALPHA.
6. Repeat step 5 on page 436 (which was done on ECMDEMO) on any subsequent client machines, additional Taskmaster Servers, and Rulerunner machines. Point them to the `datacap.xml` file so that they are aware of the additional server (which is ALPHA in this case).
 7. Ensure that both Taskmaster Servers are running. Select **Start → All Programs → Datacap → Taskmaster Server → Taskmaster Server Client**. Then click **Start**.
 8. Test that you have two working Taskmaster Servers:
 - a. Go to the first server, ECMDEMO1, and select **Start → All Programs → Datacap → Taskmaster Client**.
 - b. Select the application you want to log in to. Notice the value of Taskmaster Server(s) shows the two server names. In this example, the names are ECMDEMO1 and ALPHA. Point to two servers (Figure 12-37).

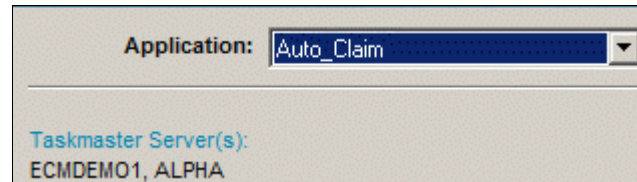


Figure 12-37 Taskmaster Servers available for connection

- c. Log in to the server. In this case, we log in to ECMDEMO1. Start a few VScan tasks to prove that it can process and run the tasks.
9. To test a failover server, complete these steps:
 - a. On ECMDEMO1, select **Start → All Programs → Datacap → Datacap Taskmaster Server**. Open **Taskmaster Server Manager**. Stop this service.
 - b. When you see a message box indicating a broken connection, log out and then log back in to the Taskmaster Client again.

This time the client states that it cannot connect to ECMDEMO1.
 - c. Click **OK**, and it connects to the secondary server ALPHA. Notice how the tasks in various states are the same as the previous login on ECMDEMO1. The reason is that both Taskmaster Servers use the same database.

You have now added additional Taskmaster Servers for failover, allowing clients and Rulerunner servers to connect to both Taskmaster Servers.

12.4.2 Load sharing between Taskmaster Servers

By load sharing two or more Taskmaster Servers, you can scale effectively in large-scale environments. To achieve this goal, update the .app application file to allow identification of the server within Rulerunner Manager:

1. Go to the application directory, and then open the application file. In this example, we go to C:\Datacap\Auto_Claims directory and open the Auto_Claim.app file.
2. Change the XML code to add an extra line for each server as shown in bold text in Example 12-7.

Example 12-7 Updated version of the Auto_Claim.app file

```
<k name="tms" ip="ECMDEMO1" port="2402" retry="3"/>
<k name="tms" ip="ALPHA" port="2402" retry="3"/>
<k name="tmsecmdemo" ip="ECMDEMO1" port="2402" retry="3"/>
<k name="tmsalpha" ip="ALPHA" port="2402" retry="3"/>
```

tms: The third line in Example 12-7 is not strictly needed. When configuring Rulerunner, the key "tms" refers to the first instance of "tms," for which this line creates another name. You only really need to define synonyms or aliases for the second and subsequent tms servers.

You must add uniquely identifiable names for the servers, leaving the original tms servers to allow the clients to connect. In our example, the uniquely identifiable names are *tmsecmdemo* and *tmsalpha*.

By adding these names, Rulerunner can also identify each server uniquely. If we use tms as we have done before, Rulerunner connects to the first tms server that it reads from the XML code and cannot differentiate between them. If the first Taskmaster Server fails, it connects to the next tms server. In this example, it is a failover from ECMDEMO1 to ALPHA.

If you want to load balance between two servers because the throughput is high, you must add the ability to assign specific tasks to specific Taskmaster Servers (or groups of Taskmaster Servers). These tasks are uniquely identified by the 'name=' tag.

If we open Rulerunner Manager on ALPHA, we see the changes that we have made. It shows the original two tms servers, including the additional tmsecmdemo1 and tmsalpha as shown in Figure 12-38.

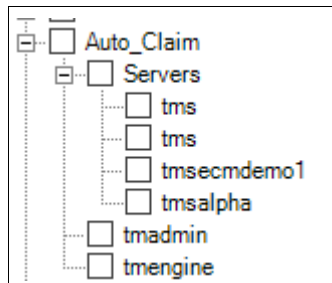


Figure 12-38 Two additional servers, tmsecmdemo1 and tmsalpha

Important: Both the tms servers and the newly named Taskmaster Servers must be present in the .app file as shown in Figure 12-7 on page 439. If the tms servers are not present, the Taskmaster Client is unable to connect to either Taskmaster Server.

We can now choose between which Taskmaster Server we assign our tasks to. Earlier in this chapter (see Figure 12-19 on page 417), we explained how to create a thread and drag tasks across it. Here, we perform this task in a slightly different way so that we can assign tasks to a thread and share the thread between two or more Taskmaster Servers on a given ratio. That is 99% of the jobs go to Taskmaster Server, and 1% go to the other server.

To create a thread and share the thread between two or more Taskmaster Servers, complete these steps:

1. Connect to the tmsecmdemo1 server:
 - a. In Rulerunner Manager, click the **Datacap Login** tab. If you need to add credentials, see 12.2.1, “Single-threaded Rulerunner” on page 402.
 - b. Click **Connect**. The text at the bottom of the window changes to reflect the connection status.
 - c. Click the **Workflow:Job:task** tab. From the list of applications, expand **Auto_Claim** → **Servers**.
 - d. Right-click **tmsecmdemo1**, which selects all the parents, and then select **Connect**. The text at the bottom of the window changes to reflect the connection to this Taskmaster Server.

For more details about the server, look in the lower right part of the ID and Settings section (Figure 12-39).

ID	
name	tmsecmdemo1
Settings	
ip	ECMDEMO1
port	2402
retry	3

Figure 12-39 Taskmaster Server details in the Rulerunner Manager

2. Connect the tadmin and tmengine databases as follows:

- a. Right-click **tadmin**, and then select **Connect**.
- b. Right-click **tmengine**, and then select **Connect**.

The text at the bottom of the window reflects the fact that you are connected to the Taskmaster Server, logged into the database, and connected to the Taskmaster Application Service.

3. Create a thread to the right of the main tree:

- a. Right-click the white space, and then select **Threads** → **Add Thread**.
- b. Select the tasks that you want to add to the thread, for example, **PageID** and **Rulerunner**.
- c. Select the parent node of the application, which is **Auto_Claim** in this case, as highlighted in Figure 12-40.

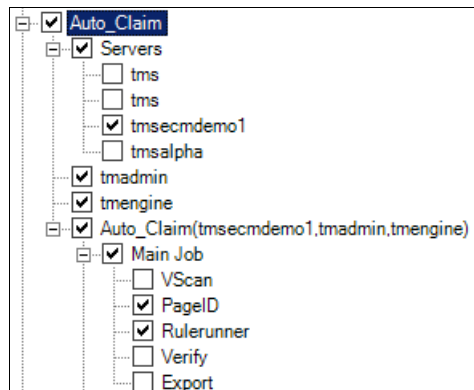


Figure 12-40 Auto_Claim is the parent node

- d. Drag this node to the thread that you just created. It populates the thread.

4. Disconnect tmengine, tmadmin, and tmsecmdemo1:
 - a. Right-click **tmengine**, and select **Disconnect**.
 - b. Repeat step a for tmadmin and tmsecmdemo1.
 - c. Clear the boxes.

The text at the bottom of the window indicates that you are only connected to the Taskmaster Application Service.

5. Select the second Taskmaster Server, which is **tmsalpha** in this case. Connect to it along with both the tmadmin and tmengine databases. Select the tasks that you need. Then select the parent node and drag it to the root of the thread (Figure 12-41).

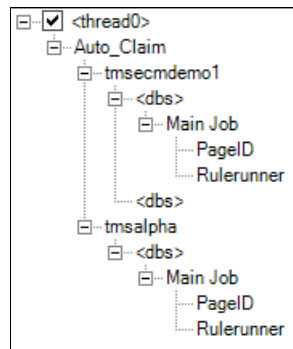


Figure 12-41 Thread setup across two Taskmaster Servers

6. Right-click the **<db>** tag in the center and select **Remove**.
7. Set priority for the server as needed.

If you click the tmsecmdemo1 node under the thread, and similarly, the tmsalpha node, you can set a priority for them in the lower right pane of the window. In this case, the priority allows you to use one Taskmaster Server more than the other. Setting one server with a higher number means that it will be used more times than the other.

For example, if we set tmsecmdemo1 with a priority of 99, and we can set tmsalpha a priority of 1. Then tmsecmdemo1 will be used 99 times out of 100. If one of these two servers fails, the other server takes on the workload and ignores any defined priorities.

8. Click the **Workflow:Job:Task** tab, and then clear the **Stop on Abort. Stop the service if a task aborts** option (Figure 12-42).

☐ Write to Debug. Log queuing activity in debug table.
☒ Mixed Queuing. Batches are selected based on priority and time stamp.
☐ Stop on Abort. Stop the service if a task aborts.

Thread Timeout: Number of seconds before threads are forced to stop.

Sleep For: Number of seconds each thread idles.

Path to Settings File:

Quattro Settings **Advanced Settings**

Figure 12-42 Clearing the Stop on Abort check box

Important: For the failover to work effectively, you must complete this step. Otherwise, all tasks cease to process across all Taskmaster Servers if a Taskmaster Server connection terminates (that is, a Taskmaster Server failure) and stops.

9. Set the restart interval for a Rulerunner service.

Rulerunner does not reconnect to a failed Taskmaster Server without restarting. If a Taskmaster Server fails, and then comes back online, restarting the Rulerunner service is required.

You can restart the service manually. Alternatively, you can lower the Restart Interval value on the **Advanced Settings** tab of the **Workflow:Job:Task** tab (Figure 12-43). A suitable value must be determined on a case-by-case basis. Restarting a server frequently might reduce throughput of the Rulerunner server.

☒ Restart service if stopped.

Restart Interval: Defines, in seconds, how often the service is stopped and restarted.

State Change Delay: Number of seconds after which the service changes state (starting/stopping).

Service Timeout: Number of seconds after which service is considered hung.

Quattro Settings **Advanced Settings**

Figure 12-43 Restart Interval setting

Figure 12-44 shows another possible configuration where a thread has been dedicated to each Taskmaster. The disadvantage of this configuration is that, if the Taskmaster Server fails, the thread becomes inactive until the Taskmaster Server comes back online and the Rulerunner service is restarted.

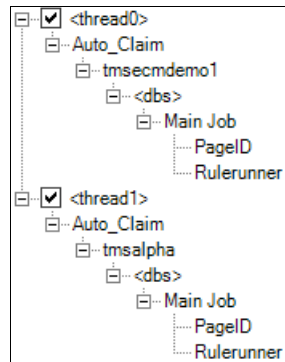


Figure 12-44 Thread running on separate Taskmaster Servers

For more information about how this process works, see “Sequential queuing mode” on page 406, “Mixed queuing mode” on page 406, and 12.3.4, “Configuring priorities and queuing” on page 425.

10. Save and test the configuration:

- a. Click **Save** to save this configuration.
- b. Click the **Datacap Login** tab, and click **Disconnect**.
- c. Start Rulerunner, and then check if processing is occurring.
- d. After the processing is occurring, check whether it works by stopping one of the Taskmaster Servers and checking that the processing still occurs.

12.4.3 Overview of Rulerunner Server and Taskmaster Server

The configuration in Figure 12-45 on page 445 shows complete redundancy if either a Taskmaster or Rulerunner server fails. Similarly, clients can connect to either Taskmaster Server. This configuration is similar to the system created in the previous sections.

The two points of failure in Figure 12-45 on page 445 are the database server and the file share. If you perform appropriate backup on these two areas, you can build a highly resilient document capture system. For information about backup, see 12.5.6, “Backing up the file share” on page 452, and 12.5.3, “Backing up the database server” on page 451.

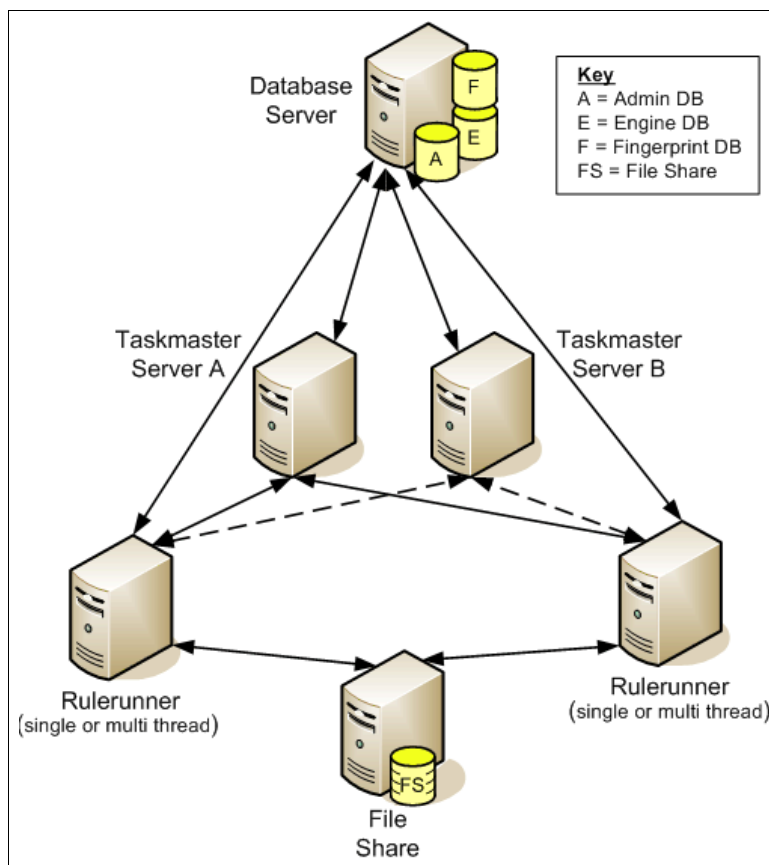


Figure 12-45 Overall configuration of the Taskmaster and Rulerunner servers

Multithread licensing: At the time of publication, an additional license, called the *Rulerunner Enterprise license*, is required to use multithreading in Rulerunner. See your IBM marketing representative for more information.

12.4.4 Scaling a thick client

The previous sections explained failover for both thick and thin clients. This section explains how to scale a thick client. The approach is to use multiple thick clients across multiple Taskmaster Servers.

To balance the load of multiple thick clients across multiple Taskmaster Servers, we must break away from the model defined earlier of the centralized `datacap.xml` and `<application>.app` files as follows:

1. For the clients to connect to additional Taskmaster Servers, make a copy of the original `datacap.xml` and `<application>.app` files. Place them in a separate shared directory.
2. In **Datacap Application Manager** on the thick client or Taskmaster Web Server, point the service path to this new copy:
 - a. Create a user group as part of the domain into which the thick and thin client users can be added.
 - b. Copy the `datacap.xml` file from the `c:\datacap\` directory to a new location. For this example, we create a folder called `c:\datacapclient1\` and copy it to this new location.
 - c. Create a copy of the `.app` file from the application that you want to scale. For this example, we copy the `Auto_Claim.app` file to a new folder of the same name under `c:\Datacapclient1\`.

You now have the following files:

- `c:\datacapclient1\datacap.xml`
- `c:\datacapclient1\auto_claim\auto_claim.app`

If you want to load balance clients in the future, consider making a minimum number of separate configurations during the initial configuration.

Additional thick clients: The additional thick clients do not have a one-to-one relationship to the `c:\datacapclientX\` configuration folder. This folder can be used by a group of many defined thick clients.

3. Open the `datacap.xml` file (Example 12-8), and then remove any unwanted applications.

Example 12-8 The datacap.xml file

```
<datacap ver="8.0">
  <app name="Auto_Claim" ref="Auto_Claim"></app>
</datacap>
```

4. Open `c:\datacapclient1\auto_claim\auto_claim.app`, and then change the XML code to reflect the primary and secondary Taskmaster Servers that you want to use (Example 12-9).

Example 12-9 The auto_claim.app file

```
<k name="tmservers">
  <k name="tms" ip="ECMDemo1" port="2402" retry="3"/>
  <k name="tms" ip="ALPHA" port="2402" retry="3"/>
</k>
```

5. Share the directory in which this file resides, which is the `\\ecmdemo1\datacap\auto_claim\` directory. The user who is running the thick client or the Taskmaster Web Server must have the appropriate permissions to access the share drive.
6. On the client (thick client or thin web server), open Taskmaster Application Manager.
7. Click the **Service** tab. For Main applications management path, enter the newly shared directory that contains the `datacap.xml` file.

The applications populate in the user interface as shown in Figure 12-46.

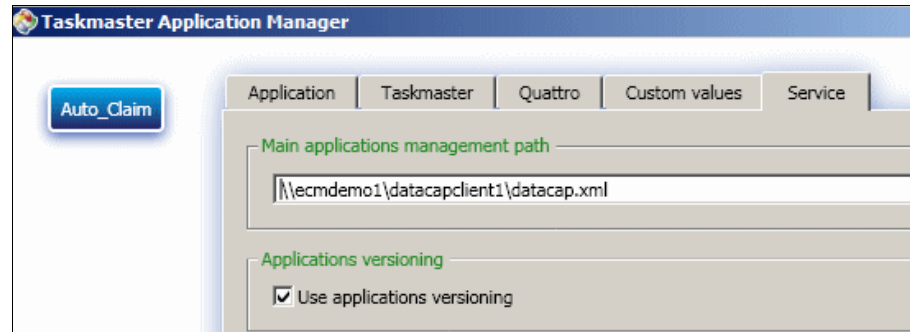


Figure 12-46 Newly shared directory on the Service tab

If the application does not populate, you can alter the registry. Run the **regedit** command, and then search for `toppath`. Alter `TopPath` to reflect the location of the new `datacap.xml` file as shown in Figure 12-47.

Name	Type	Data
(Default)	REG_SZ	(value not set)
TopPath	REG_SZ	\\ecmdemo1\datacapclient1\datacap.xml

Figure 12-47 TopPath entry

8. Connect with the client. Point it to the newly defined Taskmaster Servers that you defined, in the order in which you defined the .app file, with only the applications that you defined as available in the datacap.xml file. See Figure 12-48.

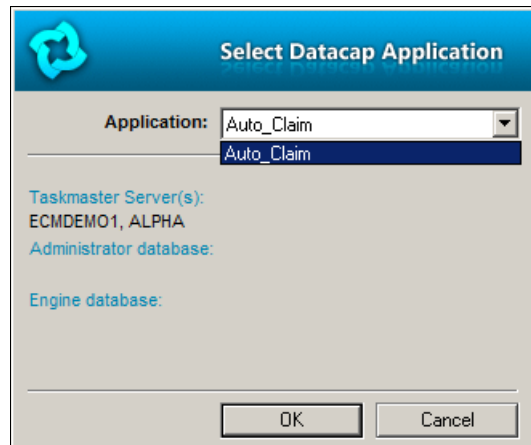


Figure 12-48 Selecting the Auto_Claim Datacap application

If the system produces an error message that the application does not specify an administrator database, verify the values on the **Taskmaster** tab in Taskmaster Application Manager. If they are blank, copy the values from the original .app file in the central server to the new copy .app file. Then, change the values to reflect the proper location of the databases.

Remember that for any changes you make at a later stage to the central .app or dataap.xml file, you must copy them to the additional directories. This way you can ensure that the changes you made to the Taskmaster XML section (see Example 12-9 on page 447) are not effected.

If you need to add additional clients to further scale out, complete these steps:

1. Copy the c:\datacapclient1\ directory and all its contents to the c:\datacapclient2\ directory.
2. Modify the .app file in the c:\datacapclient1\auto_claim\ directory to reflect the alternate Taskmaster Servers that you want to use.
3. Open **Datacap Application Manager** on the client that you want to configure, and then point it to the new directory, such as the \\ecmdemo\datacaclient2\datacap.xml directory.

12.4.5 Scaling a Taskmaster Web client

The approach to scale Taskmaster Web entails using multiple Taskmaster Web Servers across multiple Taskmaster Servers. To balance the load of multiple Taskmaster Web Servers across multiple Taskmaster Servers, Taskmaster Web works on the same principles described in 12.4.4, “Scaling a thick client” on page 445. However, rather than on the client machine, it must be carried out on the Taskmaster Webserver.

The web server inherits the Taskmaster Servers that are configured by using the `datacap.xml` and `.app` files defined in the Datacap Application Managers on the **Services** tab.

Tip: Similar to the thick client configuration, the additional web servers do not have a one-to-one relationship to a `c:\datacapclientX\` configuration folder. The `c:\datacapclientX\` configuration folder can be used by a group of many defined web servers.

Place the `datacap.xml` and `.app` files on the `c:\` drive of each web server rather than in a shared network location.

12.5 Backup and restore

Ensure that a regular backup is made of both the production and development systems. Regular backup ensures that if a failure occurs, you have a recovery point to revert to.

12.5.1 Backing up and restoring Rulerunner machines

Because Rulerunner servers are stateless (process data and push it back to the batch directory and inform the Taskmaster Server of its completion), no data is held on them. Therefore, no data is lost if the server fails. The only information that can potentially be lost is data that is in process at batch execution time.

As a result, backup is made much simpler. The Rulerunner server must be brought down gracefully when no tasks are in process. After the server is brought down, a standard mirror of the whole server can be carried out.

If a failure occurs, the server can be restored to the point at which the mirror is made. Because no processes are running at the time of creation, the system can revert to a clean Rulerunner server status. To successfully revert to a clean Rulerunner server status, downtime for Rulerunner is essential.

After an initial backup is made, subsequent backups are needed only when changes or updates are installed on the Rulerunner server such as fix packs.

12.5.2 Backing up and restoring the Taskmaster Server

The Taskmaster Server is the central Windows service that provides user authentication, workflow, and queuing (and file services for Taskmaster Web). It stores user details and the status of batches in two databases, which are the Administration and Engine databases.

Therefore, you must back up both databases to return to a preferred status. Backing up databases is well documented in many other publications and web sites. The database backup procedure is beyond the scope of this IBM Redbooks publication and, therefore, is not addressed here. Consult your database vendor documentation for further information.

In a critical environment, use a database mirror so that you always have two copies of the database.

Backing up of the Taskmaster Server ensures that you have a snapshot to restore to. You must close the Taskmaster Server gracefully, ensuring that no tasks are currently in progress.

To carry out the shutdown, follow this order:

1. Where possible, close the connected clients.
2. Close any services.
3. Shut down the Taskmaster Server.

After the system shuts down, you can carry out a standard mirror of the whole server.

If a failure occurs, the server can be restored to the point at which the mirror is made. Because no processes are running at the time of mirror creation, the system reverts to a clean Taskmaster Service status. When reverting to a mirror snapshot of the Engine database or the file share that contains document batches, any work processed between the time the mirror was created and the system failure is lost. To successfully back up the system, downtime for the system is essential.

The frequency of backup depends on the specific needs of the client.

12.5.3 Backing up the database server

Taskmaster comes with Microsoft Access as its standard shipped database format. It uses two main databases, the Engine database and the Administrator databases. A Fingerprint database is also available.

Backing up and restoring databases is documented and beyond the scope of this IBM Redbooks publication. Consult with your database vendor documentation for details.

12.5.4 Backing up and restoring the Fingerprint server

The Fingerprint Service, similar to Rulerunner, is stateless. It runs as a service and loads fingerprint details in the Fingerprint database. As a result, backing up is simpler because no fingerprints are stored on its file system.

Bring down the Fingerprint Service gracefully when no tasks are in process. After you bring it down, a standard mirror of the whole server is sufficient.

If a failure occurs, you can restore the server to the point at which the mirror is made. Because no fingerprints are loaded into memory at the time of creation, the system can revert to a clean Fingerprint Server Service. To successfully restore the system, downtime for the system is essential.

The frequency of backup depends on the specific needs of the client.

12.5.5 Backing up and restoring the IIS web server

The IIS web server serves pages to the web for users to scan, verify, and administer the system. When images are scanned, they are held on the IIS web server before they are uploaded. Apart from this process, the server is stateless. It holds no other permanent data.

Bring down the IIS web server gracefully when no tasks are in process. After the server is down, a standard mirror of the whole server is sufficient.

If a failure occurs, you can restore the server to the point at which the mirror is made. The system can revert to a clean IIS web server status. To successfully restore the system, downtime for the system is essential.

The frequency of backup depends on the specific needs of the client.

12.5.6 Backing up the file share

The file share holds all batch data and the Taskmaster application project files (if you store them there). Continuously mirroring this drive ensures that you have a fully current version of the batches. If paired with database mirroring of the Engine database, it also ensures the status of that batch.



Installation, migration, and application reuse

This chapter explains how to install the IBM Datacap Taskmaster Capture (Taskmaster) software. It includes information about application migration and reusing existing modules from one application to another.

This chapter includes the following sections:

- ▶ Installing the Datacap Taskmaster Capture software
- ▶ Migrating application from development to another environment
- ▶ Reusing applications

13.1 Installing the Datacap Taskmaster Capture software

This section focuses on the installation related to the case study in this book so that you have a reference point on how you to set up your system. For our use case, we install Taskmaster on a distributed environment with multiple machines. On each machine, we install only the components for the intended use.

For installation instructions for the single server and distributed server installation, see the *IBM Datacap Taskmaster Installation and Configuration Guide*, GC19-3232. See Appendix B of that guide for a checklist to guide you the distributed installation. This section does not repeat the instructions that are provided in the existing product documentation.

Full installation: It is possible to perform a full installation of Taskmaster on each machine. A full installation helps if debugging is required. If you installed a full version on multiple machines, ensure that Taskmaster Server, Taskmaster Web Server, and Rulerunner services are started only on the defined machines.

Table 13-1 shows the four machines that we set up for our use case.

Table 13-1 Machines in use-case scenario

Machine name	Applications	IP address
Bond	Taskmaster thin client Taskmaster thick client RV2 client	192.168.168.101
Morpheus	Taskmaster Server Taskmaster Web Server Fileserver for the project files	192.168.168.94
FileNet	FileNet system	192.168.168.127
Quantum	Rulerunner services	192.168.168.111

The first machine, called Bond, represents the client machines. It provides the Taskmaster Web interface and the Taskmaster Client. The RV2 client is used from here as well.

The second machine, called Morpheus, hosts the Taskmaster Server and the Taskmaster Web Server. The configuration of the capture solution is on the file server of this machine.

The third machine, called FileNet, is reserved for the FileNet P8 Content Manager and the FileNet P8 Business Process Manager (BPM) components.

The fourth machine, called Quantum, is a server that hosts the Rulerunner services. This machine is where the tasks are processed.

Figure 13-1 illustrates the configuration of the four physical machines.

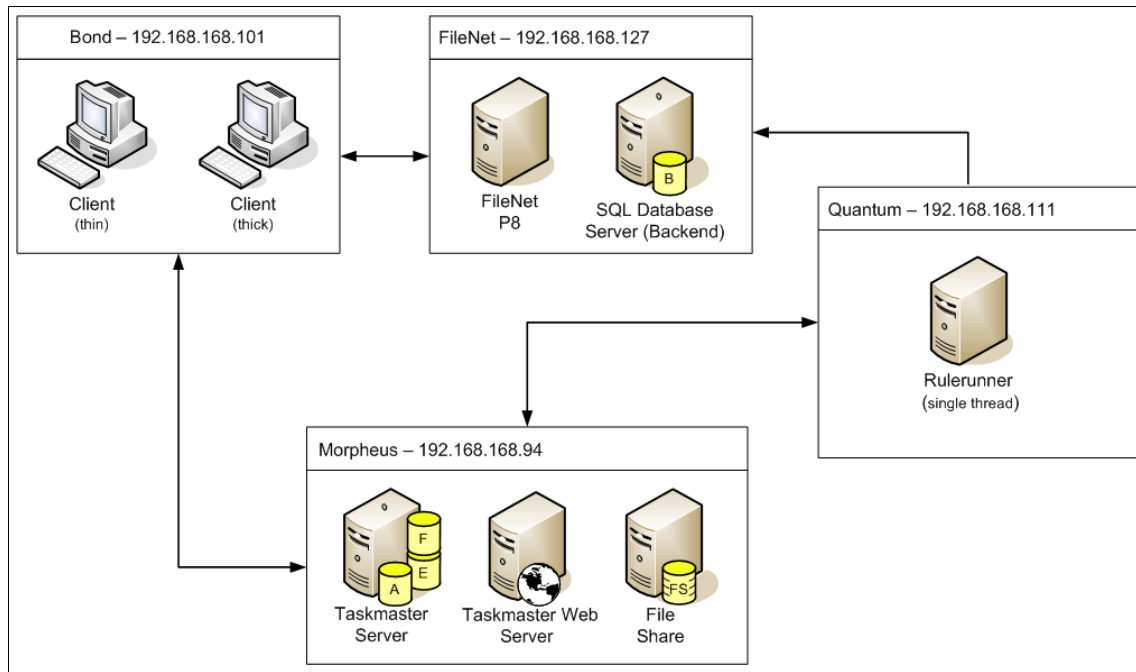


Figure 13-1 Configuration of the machines in the use-case scenario

For standard installation documentation, see the following resources:

- IBM FileNet P8 Information Center:

<http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp>

- Taskmaster documentation

IBM Datacap Taskmaster Installation and Configuration Guide, GC19-3232

13.1.1 Installing IBM Taskmaster Web client

To install the Taskmaster Web client, keep in mind the following key actions:

- ▶ Start a browser and go to the URL of the Taskmaster Web application on the Taskmaster Web Server such as the following example:

`http://server/tmweb.net/`

You must have the appropriate permissions to install Java servlets on the client machine.

- ▶ Set the security. First, select **Internet Options**, and then navigate to and click the **Security** tab.
- ▶ Set the options.

As an administrator, you want to create a script to push out these settings. In addition, you can run a tool to automatically define these settings on a workstation. The tool must be run on each thin client workstation. To find this tool, select **Start** → **Taskmaster Web Client Configuration**.

To manually set the options, make the following selections as shown in Figure 13-2 on page 457:

- For Allow ActiveX Filtering, select **Enable**.
- For Allow previously unused ActiveX controls to run, select **Enable**.
- For Allow Scriptlets, select **Disable**.
- For Automatic prompting for ActiveX controls, select **Disable**.
- For Binary script behaviors, select **Enable**.
- For Display video animation on a webpage, select **Disable**.
- For Download signed ActiveX controls, select **Prompt**.
- For Download unsigned ActiveX controls, select **Disable**.
- For Initialize and script ActiveX controls not marked as safe, select **Disable**.
- For Allow only approved domains to use ActiveX without prompt, select **Disable**.
- For Run ActiveX controls and plug-ins, select **Enable**.
- For Script ActiveX controls marked safe for scripting*, select **Enable**.

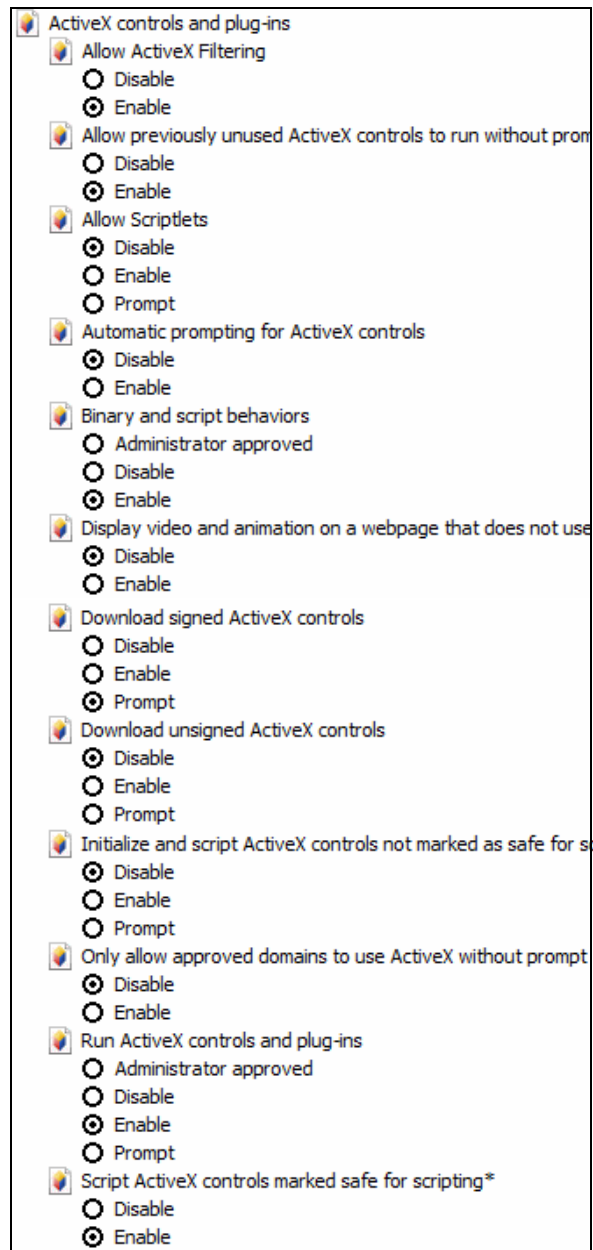


Figure 13-2 Security settings

13.1.2 Installing Taskmaster thick client

The installation program for Taskmaster installs the Taskmaster thick client. We run the installation program on our client machine, named Bond.

Follow the instructions in the installation guide. When prompted for the setup installation type, select **Custom**, and then click **Next**.

In the next window, select **Taskmaster Client** → **Applications** → **IBM Datacap Taskmaster**.

After the installation, we connect the Taskmaster Client on the Bond machine to the Taskmaster Server. Both machines must be in the same Windows domain. The client reads the configuration information from a file. Taskmaster Application Manager is the graphical user interface (GUI) for the configuration.

To access the Taskmaster Application Manager, select **Start** → **Datacap** → **Taskmaster Client** → **Taskmaster Application Manager**. On the rightmost tab, the **Service** tab, specify the location of the configuration file. To ensure a consistent configuration across all clients, you can place this configuration file on a server. Figure 13-3 shows the configuration.

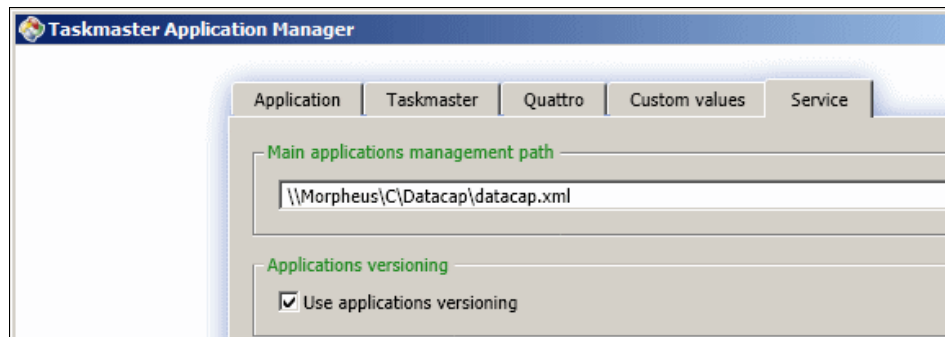


Figure 13-3 Application Manager: Defining the configuration file

13.1.3 Installing the Taskmaster Server

The installation of Taskmaster Server is described in the installation guide.

By default, the Datacap software is installed in the C:\Datacap directory. This directory contains a directory for each application. If you connect by using a network to that directory, you can use the Universal Naming Convention (UNC) notation. This directory is called the *project directory*. The project directory contains the DCO_<applicationname> Document Hierarchy (DCO) directory. Our

use case has the \\Morpheus\c\Datacap\Auto_Claim project directory and the \\Morpheus\c\Datacap\Auto_Claim\dco_Auto_Claim DCO directory.

The application-specific information is stored in the <applicationname>.app file in the project directory. Taskmaster Application Manager is the GUI for the configuration. To access the Application Manager, select **Start** → **Datacap** → **Taskmaster Client** → **Taskmaster Application Manager**. Then explore the four left-most tabs for application-specific information (Figure 13-4).

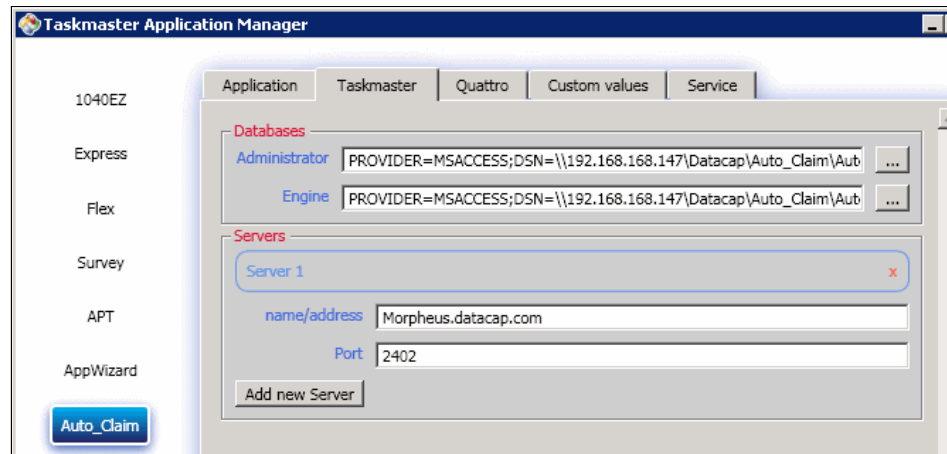


Figure 13-4 Application Manager: Setting the paths

If the project directory is on a machine, where none of the Taskmaster components are installed, configure the machine to look at the common RRS directory as shown in Figure 6-2 on page 182.

13.2 Migrating application from development to another environment

An application is made available to a client through an entry in the datacap.xml configuration file. A typical environment has just one datacap.xml file that is shared across machines. All machines are configured to use that single file. For example, in a production environment, all of the production machines share a datacap.xml file. Then, in the test system, all of the machines that are set up for the test environment share their own datacap.xml file.

The behavior of an application is typically determined by the following resources:

- ▶ Configuration files in the project directory
- ▶ Data in the databases in the project directory or subdirectories such as fingerprint data
- ▶ Application-specific rules in the DCO directory of the project
- ▶ General rules in the Datacap RRS directory

To define the project directory and DCO directory, see 13.1.3, “Installing the Taskmaster Server” on page 458.

As a good practice, use smart parameters to define the location of the project directory, the fingerprint database, and other databases, for example for data validation. To set these parameters, use the Taskmaster Application Manager as shown in Figure 13-5 on page 461.

Set up the appropriate values for the following fields as needed:

- ▶ For general:
 - Batches folder
 - Export folder
- ▶ For fingerprints:
 - Fingerprint folder
- ▶ For workflows:
 - Setup DCO
 - Rules folder
 - VScan source folder
 - Imagefix INI
 - Lookup database
 - Fingerprint database
 - Enable FPXML
 - Export database

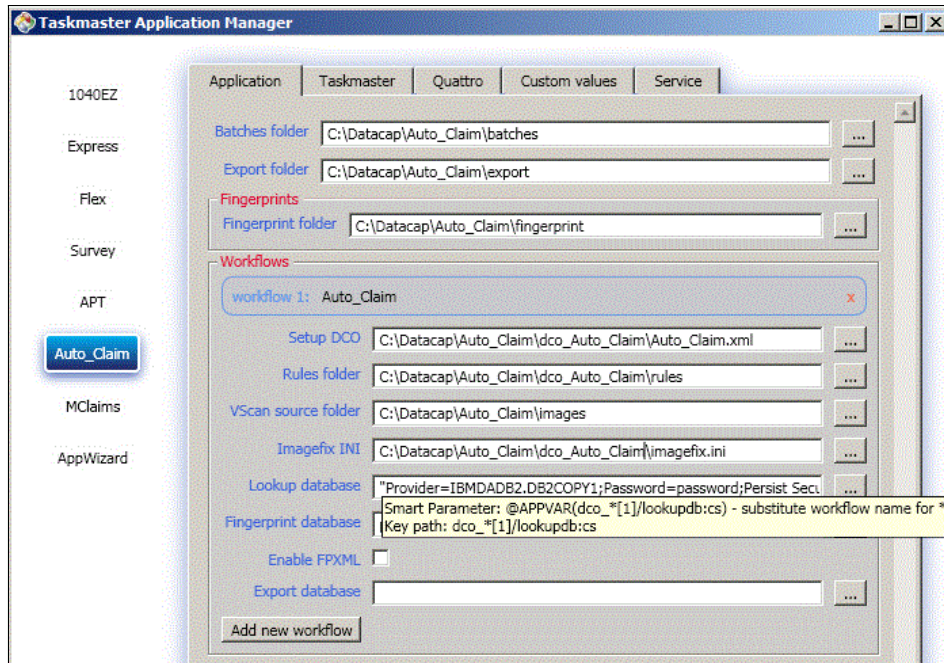


Figure 13-5 Taskmaster Application Manager: Access the environment with Smart Parameter

The Application Copy Wizard, which is built into Datacap Studio, does virtually all of the work to adjust hardcoded paths if your application has hardcoded paths. If you do not hardcode paths in your application, which is the recommended approach for creating an application, you can move your application from one environment to another environment. You move it by copying the entire application directory, except for the *.app file, the batch directory, and databases. The *.app file between one environment and another is always different. For example, the *.app file for your application in the test environment must never be copied to the production environment. The production environment maintains a unique copy of the *.app file with all of the path settings required for the production configuration.

Important: Consider using the parameters as shown in Figure 13-5. If the hardcoded paths are used within an application, you must handle them individually when the application is moved from one environment to another.

When you have access to both the source and target environments, use the Application Copy Wizard to copy the application from the source environment to the target environment.

If you do not have access from the source environment when moving the application from one environment to another, complete the following steps:

1. Copy the project directory to the target environment.
2. Adjust the parameters to point to the locations in the target environment.
3. Add an entry for the new application in the `datacap.xml` file.

For example, to migrate an application from a development environment to a production environment, copy the project directory from the development environment under the Datacap directory to the production environment. Usually, the project directory is a subdirectory of the Datacap directory. You can omit the `batches` subdirectory.

Open the Taskmaster Application Manager. Adjust the parameters to the new environment. Make sure that they point to the right databases for data validation, fingerprints, and all other required data sources.

Figure 13-5 on page 461 shows the application configuration for our use case. Specifically, you can see the path to the lookup database as it is entered and with the smart parameter notation.

To add an entry in the `datacap.xml` file for the new application, you can copy a line from an existing application and replace the name of the application and the path to the new project directory. Enter a fully qualified path to the project directory as shown in Figure 13-6.

```
<?xml version="1.0" encoding="UTF-8"?>
<datacap ver="8.0">
  <app name="1040EZ" ref="1040EZ"></app>
  <app name="Express" ref="Express"></app>
  <app name="Flex" ref="Flex"></app>
  <app name="Survey" ref="Survey"></app>
  <app name="APT" ref="APT"></app>
  <app name="Auto_Claim" ref="Auto_Claim"></app>
  <app name="MClaims" ref="MClaims"></app>
  <app name="Appwizard" ref="DStudio\Appwizard"></app>
</datacap>
```

Figure 13-6 The `datacap.xml` file with the `Auto_Claim` application added

The application is now migrated to the new environment. The client must be directed to the other environment.

To access the migrated application from the client, complete these steps:

1. Open the Taskmaster Application Manager.
2. Navigate to the **Services** tab.

3. Enter the path to the `datacap.xml` on the new server. In this example, the server is called *IntegrationTestServer* as shown in Figure 13-7.

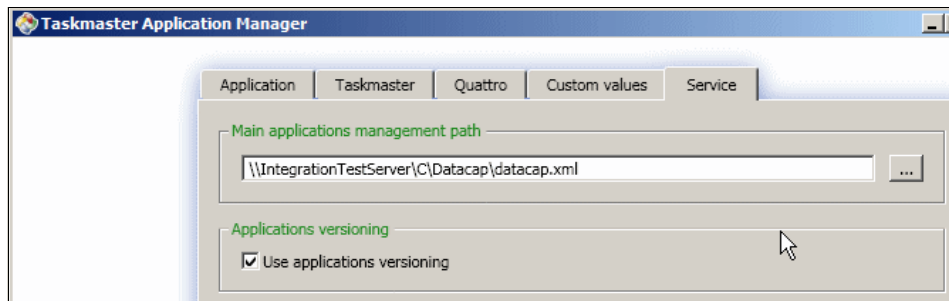


Figure 13-7 Taskmaster Application Manager: Defining the configuration file on the test server

13.3 Reusing applications

Business requirements or infrastructure requirements and constraints can change over time. For the owner of an imaging solution, it is essential to respond to this change in a timely and cost-efficient manner. Reusing an application or part of it can significantly reduce the overall application development time, increase flexibility, and help to maintain a high level of quality at minimized costs. This section explains two ways in which you can reuse an application or part of an application:

- ▶ Creating an application based on an existing one
- ▶ Enhancing a new application with existing design elements

13.3.1 Creating an application based on an existing one

You can create an application from a template that is provided with the Datacap software. You can also create an application based on an existing custom application.

To create an application, complete these steps:

1. Start the Datacap application wizard:
 - a. Open Datacap Studio.
 - b. In the application dialog box, click **Close** to access the empty Datacap Studio window.
 - c. Start the Datacap application wizard.
 - d. In the Overview window, click **Next**.

2. In the Application wizard window (Figure 13-8), select **Copy an existing RRS application**, and then click **Next**.

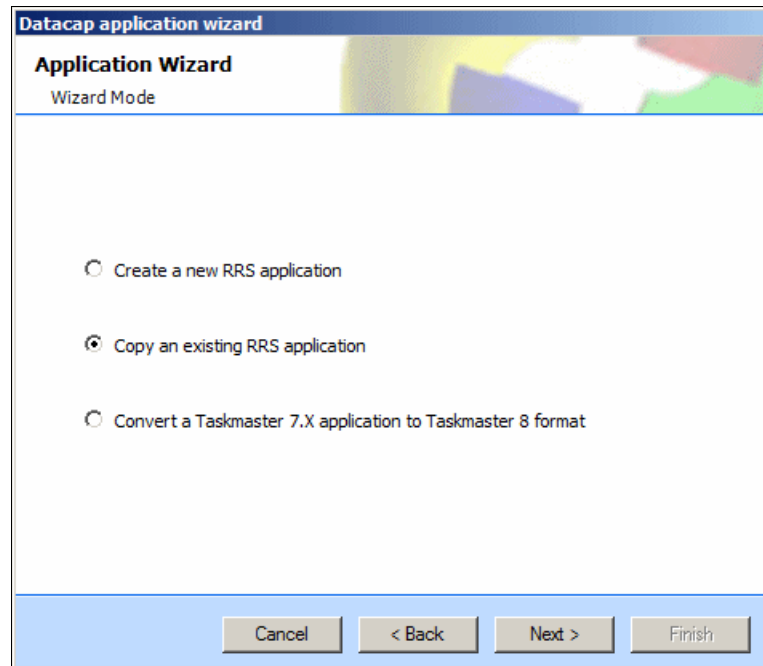


Figure 13-8 Copying an application in Application Wizard

3. In the Application Wizard—Copy and existing RRS application window (Figure 13-9), select the source application, and then enter a name for the new application. Click **Next**.

Datacap application wizard

Application Wizard
Copy an existing RRS application

Please select an application to copy from the list

Auto_Claim

Root folder on target system (Ex: \\server\Datacap):
\\Morpheus\C\Datacap

Datacap folder on Taskmaster Client workstation (Ex: C:\Datacap):
C:\Datacap

Taskmaster Web folder (Ex: \\server\Datacap\tmlweb.net):
\\Morpheus\C\Datacap\tmlweb.net

☒ Rename Copy New Name: Next_Claim_App

Cancel < Back Next > Finish

Figure 13-9 Selecting a source application and choosing name for the new one

4. In the next window, click **Finish** to create a copy of the existing application. Now, a new application is created.
5. Make your changes in the `datacap.xml` file. After the update of the file, the new application is available.

13.3.2 Enhancing a new application with existing design elements

If you want to start supporting a business process that does not have much in common with the already-supported business processes, copying a complete application might add settings and information that are not required in the new application. In this case, it is often suitable to reuse custom components from the existing application. The development environment in Datacap Studio provides the options to reuse existing actions and rule sets.

Preparing a rule set for reuse

To prepare a rule set for reuse, complete these steps:

1. In Datacap Studio, open the source application with the valuable rule set.
2. On the **Rulemanager** tab, and then on the **Rulesets** tab, right-click the rule set that you want to reuse, and select **Copy**. Figure 13-10 shows that we selected the PageID rule set to be copied.

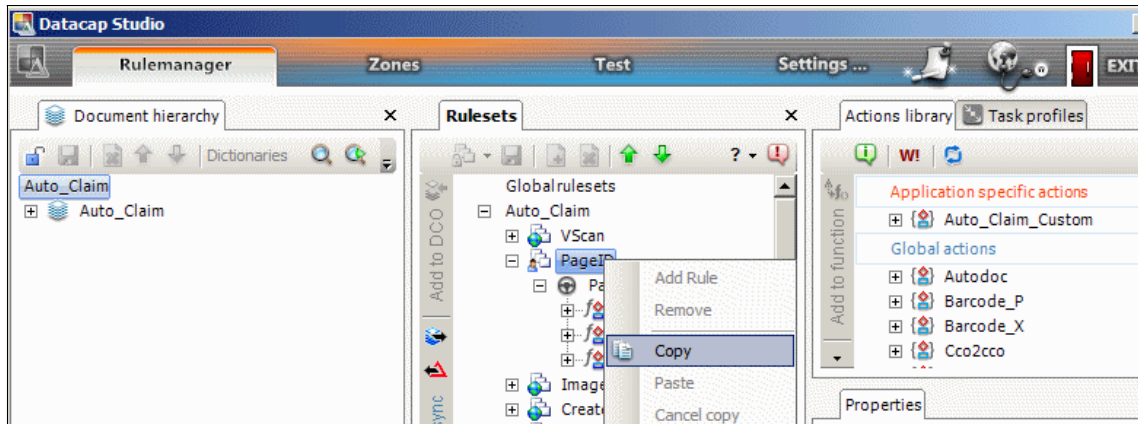


Figure 13-10 Copying an existing rule set

3. In the same frame, right-click **Global rulesets**, and then select **Paste** (Figure 13-11).

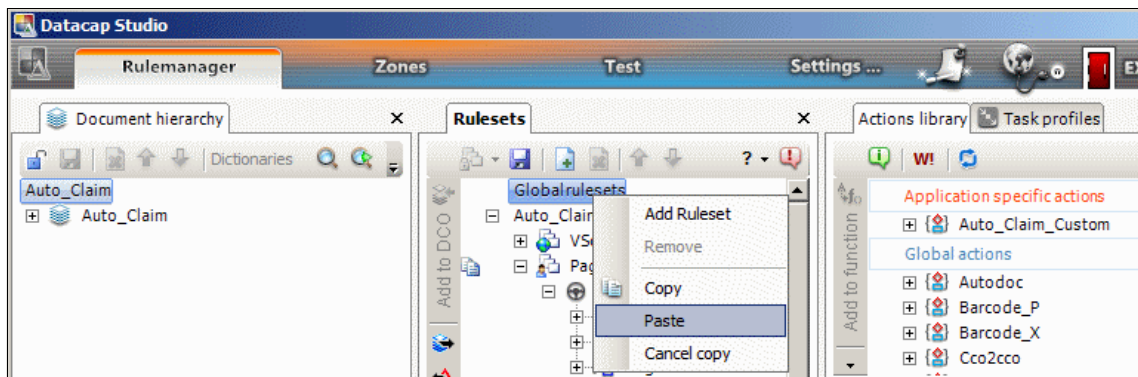


Figure 13-11 Pasting an existing rule set to a new one

4. If you are prompted to save the unlocked rule set, in the Save Ruleset dialog box (Figure 13-12), click **Yes**.

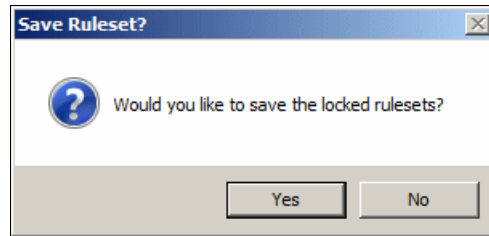


Figure 13-12 Saving a new rule set

The new rule set is now copied to the global rule sets. All actions and rules within the rule set, including parameters and their names, are copied over. Figure 13-13 shows the new PageID rule set that we copied from Auto_Claim.

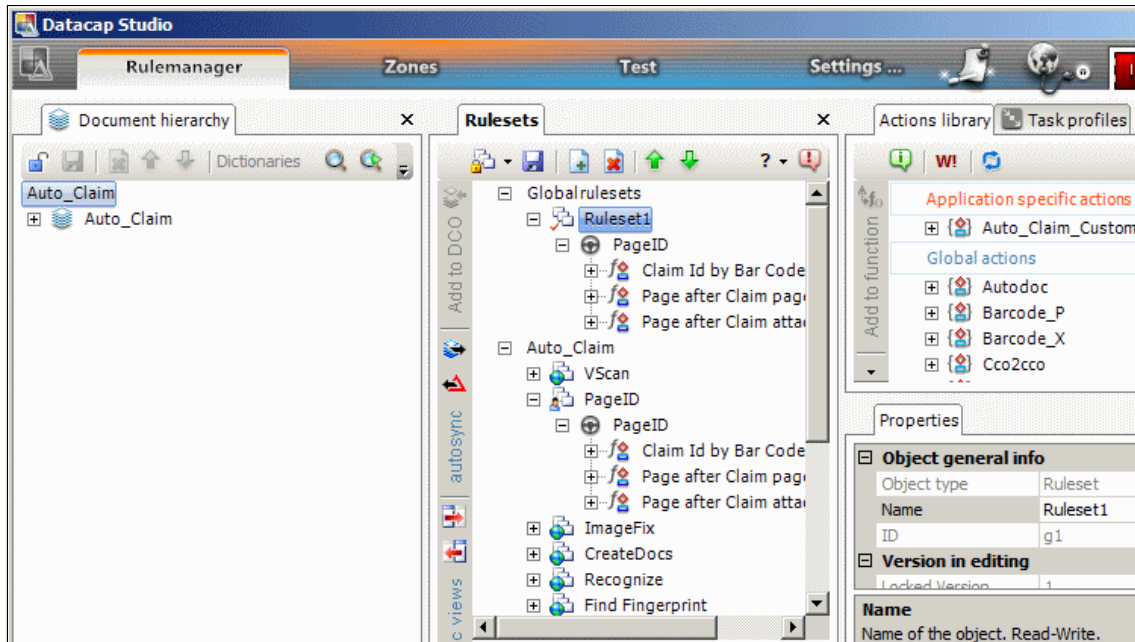


Figure 13-13 New rule set copied into Global rulesets

5. Rename the new rule set.

By default, the name of the newly copied rule set is set to Ruleset plus a sequential number. As a preferred practice, rename the generic name to a more easily understood name.

6. Save and publish the rule set.

By default, the new rule set has the “locked for editing” status. As a preferred practice, save and publish the rule set before continuing. If you omit the publishing step, you see the dialog box as shown in Figure 13-10 on page 466 when you paste the next rule set.

Applying the rule to the DCO: Remember to apply the rule to the DCO. To apply the rule, unlock the DCO for editing. Navigate to the object on which you want the rule to execute, and then click the **Add to DCO** button.

Preparing an action for reuse

The actions of an application are stored in a file, with the extension .rrx, in a subdirectory of the DCO directory. For our example, we use the \\Morpheus\C\Datacap\Auto_Claim\dco_Auto_Claim\rules directory. In general, to reuse an action, you locate the action within a rules file and then copy this file to the rules subdirectory of the DCO directory of the new application.

To reuse existing actions, complete these steps:

1. In Datacap Studio, navigate to the Actions library (Figure 13-14). Notice that there are Global actions, but no application-specific actions.

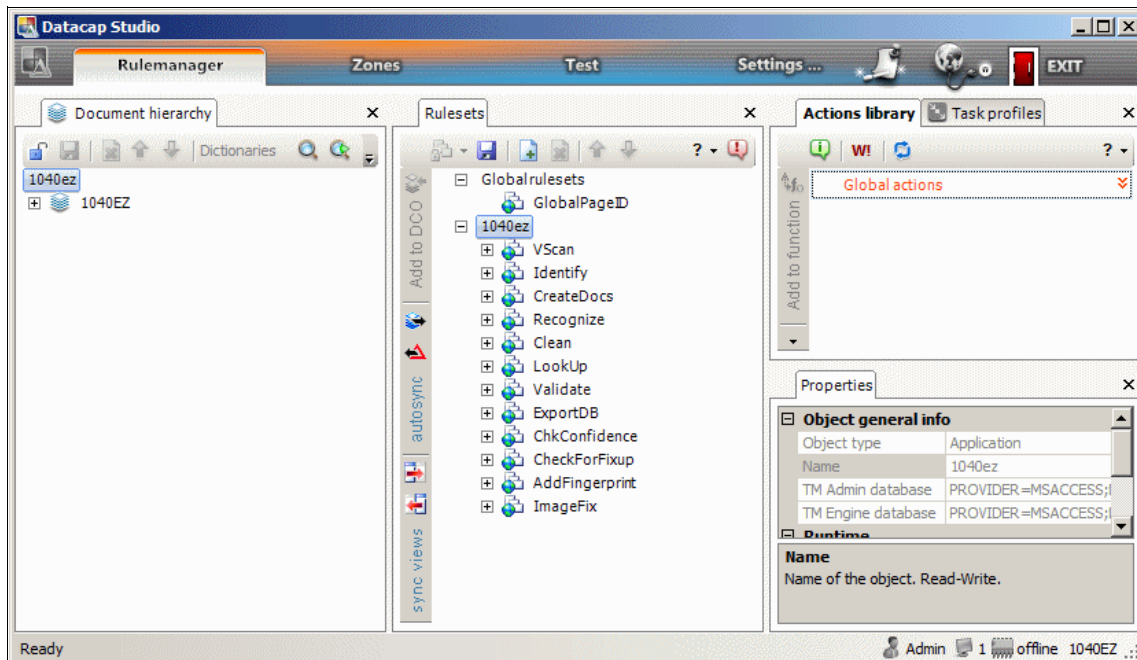


Figure 13-14 No application-specific actions

2. Open Windows Explorer, and then navigate to the rules directory.

In this example, we use the \\Morpheus\C\Datacap\Auto_Claim\dco_Auto_Claim\rules directory. We access the directory locally on Morpheus.

3. Right-click the file with actions that you want to use in the new application, and then select **Copy**. For this example, we use the auto_Claim_Custom.rrx file (Figure 13-15).

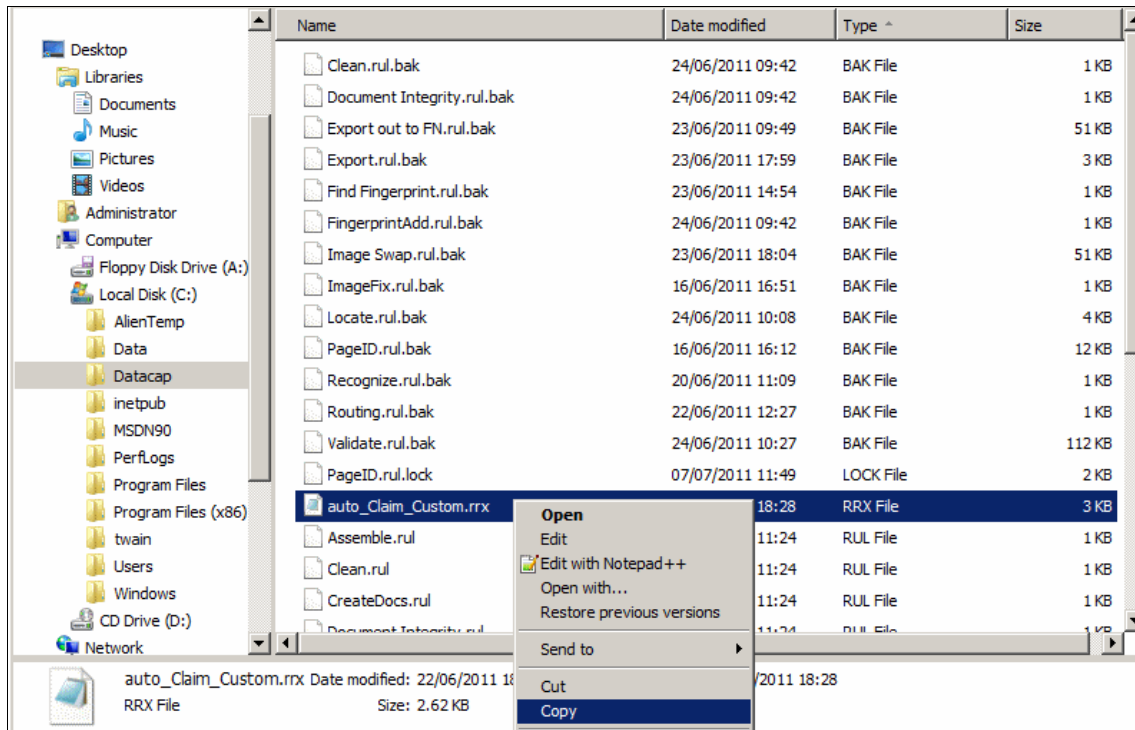


Figure 13-15 Copying the actions file

4. In Windows Explorer, in the subdirectory of the new application, right-click and select **Paste** to move to the rules.

In this example, we use the \\Morpheus\C\Datacap\1040ez\dco_1040ez\rules directory. Again, we access the directory locally on Morpheus as shown in Figure 13-16.

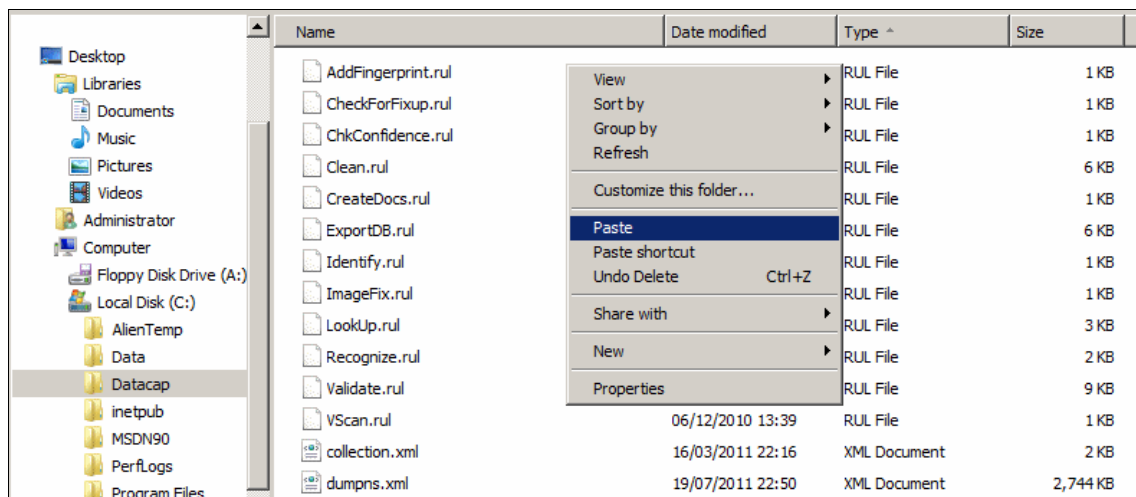


Figure 13-16 Pasting the actions file in the target directory

The rules subdirectory of the new application now contains the auto_Claim_Custom.rrx file as shown in Figure 13-17.

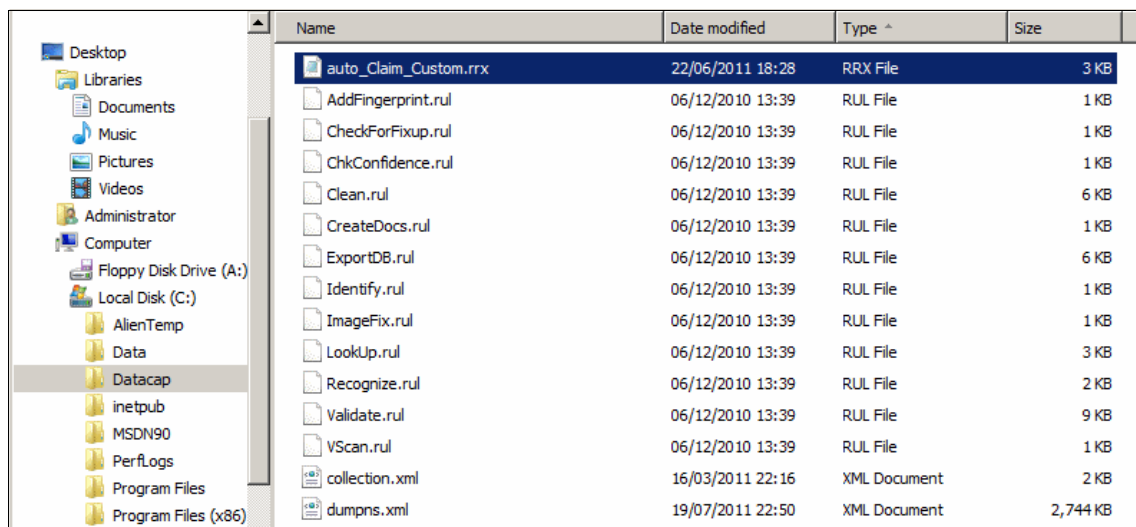


Figure 13-17 Actions file pasted

Back in Datacap Studio, nothing has changed in the action library yet.

5. To load the just added actions into the Actions library, click the **Refresh actions library view** icon to refresh the view. This icon is the rightmost icon in the icon bar on the **Actions library** tab (Figure 13-18). After the refresh, you see the new set of actions called “Application-specific actions.”

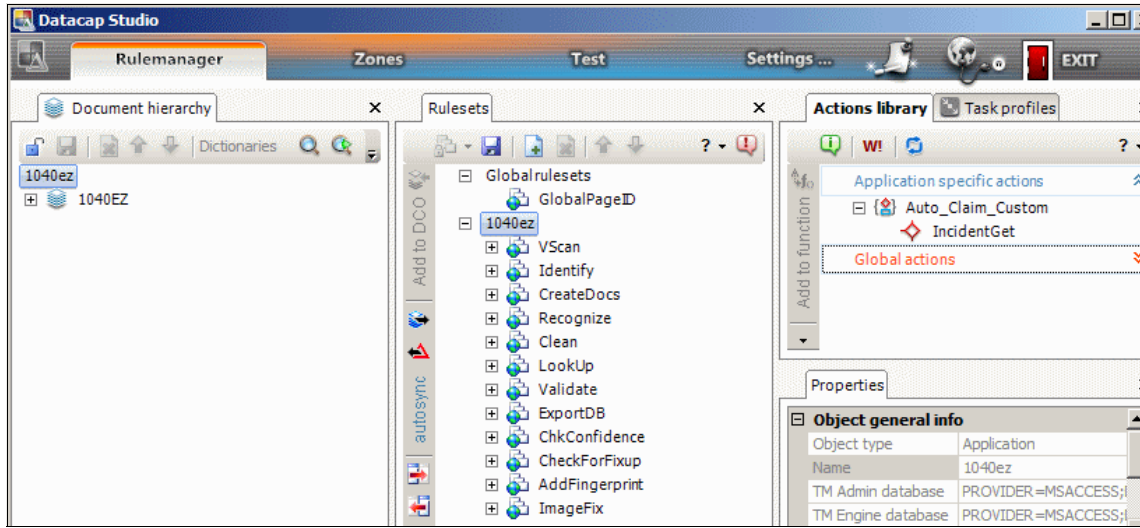


Figure 13-18 Viewing newly added actions after refreshing the view of actions

Now, the actions are available, and you can add them to rules.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *ACI Worldwide's BASE24-eps V6.2: A Supplement to SG24-7268*, REDP-4338
- ▶ *Disaster Recovery and Backup Solutions for IBM FileNet P8 Version 4.5.1 Systems*, SG24-7744
- ▶ *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547
- ▶ *IBM FileNet P8 Platform and Architecture*, SG24-7667
- ▶ *IBM High Availability Solution for IBM FileNet P8 Systems*, SG24-7700
- ▶ *Introducing IBM FileNet Business Process Manager*, SG24-7509

You can search for, view, download or order these documents and other Redbooks, Redpapers, web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

The publication *Taskmaster Application Development Guide*, SC19-3251, is also relevant as a further information source. This guide provides extensive tutorials for Taskmaster design. It is included in the Production Imaging Edition product documentation. Read this guide before you design and implement any capture system.

Online resources

These websites are also relevant as further information sources:

- ▶ Document Image and Capture main web site:
<http://www.ibm.com/software/ecm/document-imaging/>
- ▶ Production Imaging Edition main web site:
<http://www.ibm.com/software/data/content-management/production-imaging/>
- ▶ Datacap Taskmaster Capture main web site:
<http://www.ibm.com/software/ecm/datacap/taskmaster/>
- ▶ FileNet Content Manager main web site:
<http://www.ibm.com/software/data/content-management/filenet-content-manager/>
- ▶ FileNet Content Manager main web site:
<http://www.ibm.com/software/data/content-management/filenet-business-process-manager/>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redbooks

Implementing Imaging Solutions with IBM Production Imaging Edition and IBM Datacap Taskmaster Capture

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Implementing Imaging Solutions with IBM Production Imaging Edition and IBM Datacap Taskmaster Capture

Solution overview, capabilities, and system architecture

Solution planning, design, and implementation

Practical information with a use-case example

Organizations face many challenges in managing documents that they need to conduct their business. IBM Production Imaging Edition V5.0 is the comprehensive product that combines imaging, capture, and automation to provide the capabilities to process and manage high volumes of document imaging over their entire life cycle.

This IBM Redbooks publication introduces Production Imaging Edition, its components, the system architecture, its functions, and its capabilities. It primarily focuses on IBM Datacap Taskmaster Capture V8.0, including how it works, how to design a document image capture solution, and how to implement the solution using Datacap Studio. Datacap Studio is a development tool that designers use to create rules and rule sets, configure a document hierarchy and task profiles, and set up a verification panel for image verification.

This book highlights the advanced technologies that are used to create dynamic applications, such as IBM Taskmaster Accounts Payable Capture. It includes an in-depth walkthrough of the dynamic application, Taskmaster Accounts Payable Capture, which provides invaluable insight to designers in developing and customizing their applications.

In addition, this book includes information about high availability, scalability, performance, and backup and recovery options for the document imaging solution. It provides known best practices and recommendations for designing and implementing such a solution.

This book is for IT architects and professionals who are responsible for creating, improving, designing, and implementing document imaging solutions for their organizations.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks